НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-методичний комплекс
«Інститут післядипломної освіти»

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

"___" _____ 2020 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності 123 «Комп'терна інженерія»

з напряму підготовки 6.050102 «Комп'ютерна інженерія»
(код і назва)

на тему: Розподілена система моніторингу стану об'єктів охорони

Виконав: слухач IV курсу, групи ЗКІ-зп71
(шифр групи)

Соломко Володимир Вікторович
(прізвище, ім'я, по батькові) (підпис)

Керівник _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю Клятченко Ярослав Михайлович ___ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань
Слухач _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-методичний комплекс
«Інститут післядипломної освіти»

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.П.Тарасенко
(підпис)

"___" _____ 20___ р.

ЗАВДАННЯ
на дипломний проект слухачу

_____
(прізвище, ім'я, по батькові)

1. Тема проекту: Розподілена система моніторингу стану об'єктів охорони,

керівник проекту _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом НМК „ІПО" КПІ імені Ігоря Сікорського від «__» _____ 201___ р. №____

2. Термін подання слухачем проекту: дивитися технічне завдання

3. Вихідні дані до проекту: титульна сторінка, програма, технічне завдання (4 сторінки), пояснювальна записка (52 сторінки)

4. Зміст пояснювальної записки: перелік скорочень, умовних позначень та термінів, вступ, аналіз існуючих розподілених програмних засобів моніторингу та обґрунтування теми дипломного проекту, аналіз технологій для розробки розподілених систем моніторингу, структура системи та опис роботи модулів, тестування системи та аналіз результатів, висновок, список використаної літератури.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): презентація, структурні схеми, розташування модулів програми,

загальна схема проекту, схема алгоритмів, основний маршрут користувача, система моніторингу.

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	24.04.2020	
2.	Розроблення та узгодження технічного завдання	30.04.2020	
3.	Аналіз існуючих рішень	08.05.2020	
4.	Підготовка матеріалів першого розділу дипломного проекту	15.05.2020	
5.	Підготовка матеріалів другого розділу дипломного проекту	22.05.2020	
6.	Підготовка графічної частини дипломного проекту	29.05.2020	
7.	Оформлення документації дипломного проекту	05.06.2020	
8.	Попередній огляд	12.06.2020	

Слухач _____ (підпис) _____ (ініціали, прізвище)

Керівник проекту _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту

АНОТАЦІЯ

Об’єкт розробки – створення розподіленої системи моніторингу стану об’єктів охорони.

Комп’ютерна система дозволяє: відслідковувати спрацювання тривожних сповіщувачів по периметру охоронюваного об’єкту, автоматично виявляти причину спрацювання, надсилати відповідні повідомлення користувачу.

В ході розробки :

- проведено аналіз різних методів виявлення проникнення на охоронюваний об’єкт;
- сформульовано вимоги розподіленої комп’ютерної системи;
- розроблено систему моніторингу охоронюваних об’єктів на основі мікросервісної структури;
- залучено методи розподіленої обробки запитів;
- здійснено навантажувальне тестування.

Система може надавати єдиний програмний інтерфейс, який дозволить незалежно впроваджувати, а також розробляти різні продукти з метою спостереження за об’єктами охорони.

Ключові слова:
РОЗПОДІЛЕНА СИСТЕМА МОНІТОРИНГУ СТАНУ ОБ’ЄКТІВ ОХОРОНИ.

ANNOTATION

The object of development is the creation of a distributed system for monitoring the condition of protected objects.

The computer system allows: to monitor the operation of alarm detectors around the perimeter of the protected object, automatically detect the cause of the operation, send appropriate messages to the user.

During development:

- the analysis of various methods of detection of penetration on the protected object is carried out;
- formulated the requirements of a distributed computer system;
- the system of monitoring of protected objects on the basis of microscopic structure is developed;
- methods of distributed request processing are involved;
- load testing was performed.

The system can provide a single software interface that will allow you to independently implement and develop different products to monitor protected objects.

Keywords:

DISTRIBUTED SYSTEM OF MONITORING THE CONDITION OF SECURITY OBJECTS.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.4671.002 ТЗ	Розподілені програмні	4		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Технічне завдання			
	А4	ІАЛЦ.4671.003 ТП	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.4671.004 ПЗ	Розподілені програмні	52		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.001 ОА		
Розробив		Соломко В.В.			Розподілені програмні засоби. Засоби підвищення продуктивності технічних засобів охорони. ***Опис альбому***	Літ.	Аркуш / Аркушів
Перевірив							
Н. контроль						КПІ ім. Ігоря Сікорського, ЗКІ-зп71	
Затвердив							

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.4671.005 Д1	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Розташування модулів			
			програм			
			Схема структурна			
	А4	ІАЛЦ.4671.006 Д2	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Загальна схема проекту			
			Схема структурна			
	А4	ІАЛЦ.4671.007 Д3	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Основний маршрут			
			користувача			
			Схема алгоритму			
	А4	ІАЛЦ.4671.008 Д4	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Система моніторингу			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Розподілені програмні засоби моніторингу об’єктів охорони. Засоби підвищення продуктивності засобів технічної охорони».

Галузь застосування: моніторинг об’єктів охорони.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення розподіленої системи на основі якої створюють різні підсистеми для моніторингу, обліку об’єктів охорони та їх керування.

4. ДЖЕРЕЛО РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Обробка надходження сигналів зі сповіщувачів для подальшої створення їх моделі в базі даних;
- Сповіщення про виняткові ситуації;
- Розподілена архітектура;
- Відмовостійкість та здатність витримувати високі навантаження;

5.2. Вимоги до апаратного забезпечення

- Процесор: мін. 2-х ядерний процесор;
- Оперативна пам’ять: мін. 8 Гб;
- Наявність доступу до мережі Інтернет (Ethernet);
- Зв’язок GSM, GPRS;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Будь-який пристрій зі здатністю відправляти запити та дані стосовно спрацювання сигналів оповіщувачів.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	24.04.2020
2.	Розроблення та узгодження технічного завдання	30.04.2020
3.	Аналіз існуючих рішень	08.05.2020
4.	Підготовка матеріалів першого розділу дипломного проекту	15.05.2020
5.	Підготовка матеріалів другого розділу дипломного проекту	22.05.2020
6.	Підготовка графічної частини дипломного проекту	29.05.2020
7.	Оформлення документації дипломного проекту	05.06.2020
8.	Попередній огляд	12.06.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.467100.004 ПЗ	Розподілені програмні	52		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Пояснювальна записка			
	А4	ІАЛЦ.467100.005 Д1	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Розташування модулів			
			програми			
			Схема структурна			
	А4	ІАЛЦ.467100.006 Д2	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Загальна схема проекту			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.003 ТП			
Розробив		Соломко В.В.			Розподілені програмні засоби. Засоби підвищення продуктивності технічних засобів охорони	Літ.	Аркуш	Аркушів
Перевірив								
Н. контроль					Відомість технічного проекту	КПІ ім. Ігоря Сікорського, ЗКІ-зп71		
Затвердив								

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467100.007 Д3	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Основний маршрут			
			користувача			
			Схема алгоритму			
	А4	ІАЛЦ.467100.008 Д4	Розподілені програмні	1		
			засоби. Засоби підвищення			
			продуктивності			
			технічних засобів охорони			
			Система моніторингу			
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – програмне забезпечення;

JSON (JavaScript Object Notation) – текстовий формат обміну даних, оснований на JavaScript;

TCP (Transport Control Protocol) – протокол транспортного рівня передачі даних в моделі OSI;

HTTP (Hyper Text Transfer Protocol) – протокол прикладного рівня передачі даних в моделі OSI;

BSON (Binary JSON) – комп’ютерний формат обміну даними;

RFC (Request for Comments) – документ із серії пронумерованих інформаційних документів Інтернету, що містить технічні специфікації та стандарти.

TLS (Transport Layer Security) – криптографічний інтернет-протокол;

GSM (Global System Communications) – глобальна система мобільного зв’язку

ВСТУП

Зазвичай, проблеми злагодженої роботи в розподілених системах існують на усіх рівнях програмного забезпечення (ПЗ), а також обладнання. Також це стосується веб-систем, які додають труднощів з мережами. Тому слід зазначити, що існує дуже багато підходів для вирішення цих проблем та немає жодного, щоб підходив усім, тобто універсального.

Для напрямку, такого як «Охорона» можна сказати, що критичним питанням постає миттєвість, але класична існуюча архітектура «клієнт-сервер» не вирішує це питання. Загалом, Всесвітня мережа початково призначалася тільки для передачі-прийняття (обміну) документами, тому відсьогодні з метою збільшення функціональності та інтерактивності застосунків щороку зростає їх складність, а нові розроблювальні підходи збільшуються щомісячно.

Слід зазначити, що при довгострокових (довгоочікуваних) операціях найпоширеніші технології для реалізації серверів можуть працювати не так ефективно, що в свою чергу витрачає ресурси процесорів даремно.

Більшість корпорацій світу (а саме найбільші) доволі давно зіткнулися з такими проблемами та проводили відповідну роботу щодо їх подолання. На той час, це були тільки їхні проблеми, але з появою нових учасників, з’явилося безліч технологій, підходів, мов програмування та стандартів для вирішення більшості поставлених завдань. Головне, щоб використовували їх прваильно.

1. АНАЛІЗ РОЗПОДІЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ МОНІТОРИНГУ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Причини використання охоронних систем

Системи охоронного призначення – це автоматизований комплекс, необхідний для забезпечення охорони різних об’єктів майна (в т.ч. автомобілі, водний транспорт, будівлі та прилегла територія, сейфи та інше).

Вони призначені для того, щоб виявляти несанкціонований доступ на об’єктах охорони та використовуються з метою захисту від проникнення, крадіжок та псування майна [1].

Встановлення охоронного призначення на об’єктах забезпечує зниження рівня проникнення на об’єкти та здійснення на них крадіжок [2].

Існує багато охоронних систем, які додатково забезпечують захист як від пожеж та і від залиття водою, вони називаються комбінованими системами. Вони є досить гнучкими, тому можуть поєднуватися з іншими підсистемами та використовувати в один той самий час безліч різних типів сенсорів, зв’язку, сповіщувачів та інше. Отже можна сказати, що охоронні системи є досить потужними інструментами, які можуть модифікуватися, удосконалюватися та розвиватися в різних напрямках і різними способами, все це залежить від самого об’єкту охорони, тобто від його вимог та складності.

У ході роботи даного дипломного проекту було розроблено систему, яка забезпечує повне перекриття технічними засобами охорони певного об’єкту. Це дозволяє виявляти та перекривати вразливі місця проникнення на об’єкт охорони.

1.2. Огляд принципів роботи охоронних систем

Загалом до охоронних систем ми можемо віднести сюди будь-які технічні системи, які збільшують безпеку певного об’єкту охорони.

Із розвитком своїх можливостей систем охоронного призначення, а точніше їхньої мініфакції, зросту їх можливостей, пропускної здатності та надійності, почав стрімко збільшуватися ринок охоронних систем. Тому, дуже багато охоронних фірм, поліція охорони та інші замовники охоронних послуг користуються засобами технічної охорони, здійснюють монтажі на своїх офісах, складах, квартирах, будинках, автомобілях та інше.

Слід зазначити, що охоронна сигналізація має основні загальні компоненти: пристрій виявлення, блок керування та пристрій звітності. Для виявлення охоронної системи в захищеному приміщенні встановлюється пристрій виявлення. За забезпечення живлення датчиків відповідає блок керування, який приймає та оцінює сигнали від датчиків. Для подачі сигналу про тривогу можуть бути як звукові так і візуальні пристрої, які встановлені на об’єкті охорони. Кожен із пристроїв може бути інтегрованим та розробленим у різні прикладні пристрої, а також системи в залежності від потреби безпеки та відповідних затрат.

Охоронні системи можна класифікувати [3] наступним чином:

- за взаємодією із загрозою: активні та пасивні (активні – призначені для того, щоб запобігти проникненню на об’єкт охорони або відкриття дверей, вікна, сейфу та інше, пасивні – це комплекс дій та засобів, які спрямовані на пряме залучення уваги власника об’єкту або охоронної служби, в даному дипломному проекті буде використовуватися цей тип);
- за способом передачі інформації: провідні, бездротові, GSM-мережі.

Слід зазначити, що в окремих випадках можливе комбінування як бездротових із дротяними, а також пасивних з активним системами охорони.

Розглянемо більш детально охоронні системи за способом передачі інформації.

1) Провідні:

- Аналогові – даний спосіб передачі використовує пристрої прийому-контролю (ППК), він відслідковує стан шлейфу сигналізації, який напряму підключений до датчиків. У разі спрацювання якогось датчику, в шлейфі струм міняється і ППК це фіксує. Одною із проблем являється те, що тяжко визначити, який із датчиків по периметру спрацював, оскільки всі датчики на струм впливають однаково.

- Адресні – при використанні адрес можна вирішити проблеми, які виникають при аналогових.

2) Бездротові – використовуючи цей спосіб сповіщувачі (датчики) передають інформацію в радіосигналах на приймальну станцію.

- Без зворотного зв’язку – можна назвати лише недоліки, оскільки є дуже велика ймовірність перешкоджання передачі даних;

- Зі зворотнім зв’язком – даний спосіб дає можливість підтримувати постійний моніторинг.

3) Через GSM-мережі – даний спосіб використовується для того, щоб інформувати власника об’єкту охорони, а також передавати сигнал «Тривоги» на пульт централізованого спостереження охоронної служби;

4) Через протоколи Wi-Fi – даний спосіб подібний до GSM-мережі, однак інформація, що надходить до кінцевого користувача передається за допомогою відповідного ресурсу через мережу Інтернет чи мобільного застосунку. В даному дипломному проекті буде використовуватися цей спосіб із комбінованим використанням ресурсу Інтернет та застосунку, та реалізовано технологією Progressive Web Application (PWA);

Розглянемо більш детально охоронні системи за способом сповіщення про «Тривогу».

1) Центральна станція – при вторгненні на об’єкт охорони, сповіщення автоматично передається до агентства, яке слід називати центральною станцією. Де відповідні оператори завжди здійснюють нагляд, запис та реагують на сигнал «Тривога». Після чого, оператор терміново направляє слідчих та телефонує в поліцію;

2) Охоронна фірма – у цьому випадку всі пристрої та схеми виявлення підключені до постійного приймального обладнання на центральній контрольній станції;

3) Поліцейська станція – вона підключена до охоронної сигналізації, яка складається із захисних пристроїв та ланцюгів, які в свою чергу, підключені через блок керування до обладнаного поліцейського відділу. При вторгненні на об’єкт охорони поступає звуковий або візуальний сигнал, що сигналізує поліцейський відділ. Але слід зазначити, що експлуатація виконується лише під контролем власника об’єкта охорони;

4) Локальна сигналізація – даний тим охорони подає виключно звуковий або візуальний сигнал «Тривога» з метою відлякування зловмисників від

об'єкту охорони. Тобто за це відповідає лише власник об’єкта охорони і як наслідок, немає гарантії, що дана система стоїть увімкнена;

5) Житлова охоронна сигналізація – вона складається, як правило, з пристроїв та схем, які підключені через блок керування до зондового чи візуального пристрою, котрий може мати віддалене підключення до поліцейської, центральної, власної чи станції моніторингу.

1.3. Аналіз існуючих охоронних систем

При суттєвому розвитку технологій, як правило, розпочали вдосконалюватися алгоритми машинного навчання, які мають своє місце у сфері охоронної діяльності. Технологічні компанії використовують у своїй діяльності комбіновані системи, оскільки вони є найефективнішими та більш функціональними.

На українському ринку охоронних технологій має досить високе місце компанія AJAX System, яка здійснила досить великий внесок щодо модифікації технічних засобів охоронного призначення, варто назвати лише, те, що вона розробила власну систему, яка дозволяє під’єднати до 150-ти пристрої одночасно [4]. Для втілення в реальність даного проекту, компанія розробила власну операційну систем (ОС) реального часу для хабів, які мають здатність робити телефонні дзвінки та надсилати повідомлення. Це в свою чергу привело до то, що потрібно було створити власний радіо протокол передачі даних, який зміг би підтримувати відразу ж 150 з’єднань із пристроями (дивитися на рисунок 1.1.), враховуючи те, щоб не втратити в затримці обробки даних та швидкодії. Також, слід зазначити, що здійснена велика робота в частині побудови алгоритмів обробки даних від сенсорів (чи людина перебуває на охоронюваному об’єкті чи тварина).

З метою зменшення кількості хибних спрацювань сигналів «Тривога», алгоритми аналізують потік даних відразу з декількох сенсорів, сюди відноситься і спектральний аналіз.

Рисунок 1.1.

Компанія AJAX Systems створила додатки, за допомогою яких, власник охоронюваного об’єкту може слідкувати зі свого телефону за обстановкою на власному об’єкті та бачити в повній мірі, що там відбувається. Також віддалено ставити та знімати з охорони сигналізацію.

Провідні охоронні компанії останнім часом працюють із залученням хмарних технологій (дивитися на рисунок 1.2.), що дозволяє створювати та вдосконалювати гнучкі системи із високою швидкістю розгортання інфраструктури, використовуючи готові рішення для моніторингу. Слід зазначити, що це вносить значний внесок в зменшення витрат, оскільки покупець платить стільки, скільки й використав.

Хмарні технології надають можливість повністю абстрагуватися від інфраструктурного шару.

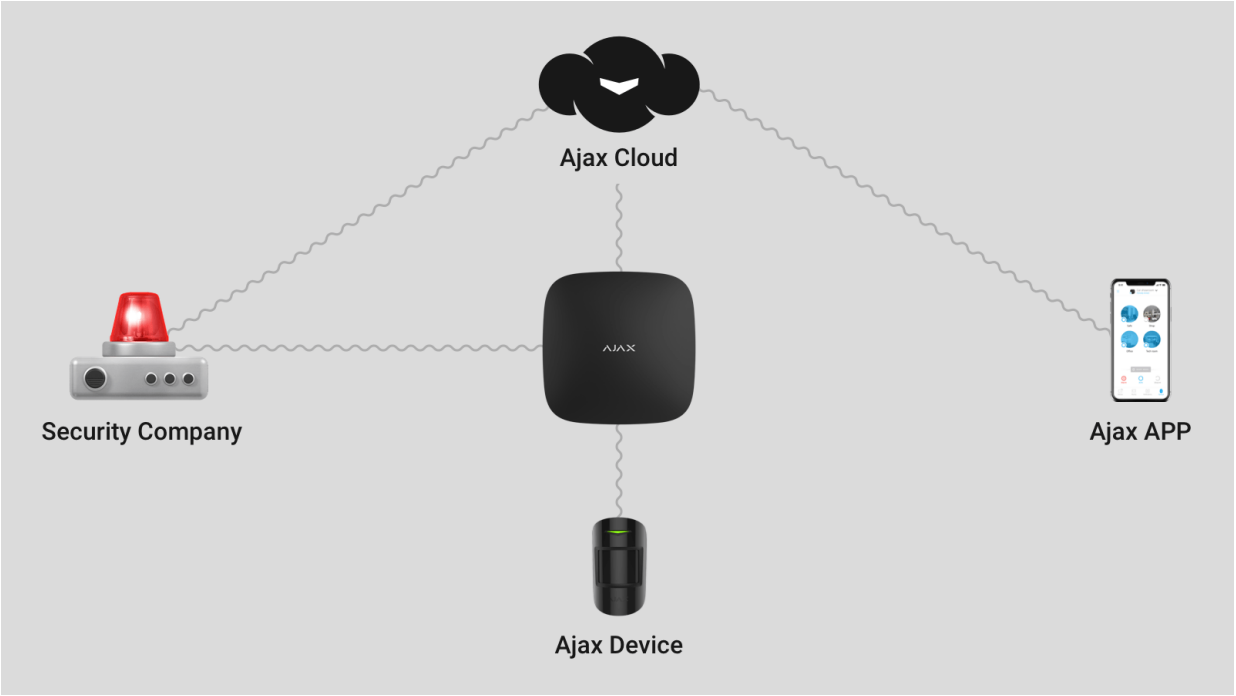

Рисунок 1.2.

В даному дипломному проекті буде використано віртуальний виділений сервер (ВВС) від Digital Ocean (рисунок 1.3.).

Рисунок 1.3.

Загалом, віртуальний виділений сервер запускає особисту копію операційної системи (ОС), отже користувачі зможуть мати доступ на правах супер користувача до цього екземпляру операційної системи та матимуть можливість встановлення практично будь-якого програмного забезпечення (ПЗ), яке працюватиме на цій ОС.

Вони більш дешеві, чим еквівалентний фізичний сервер. Але продуктивність може бути нижчою так, як вони розділяють основне фізичне обладнання з іншими Virtual Private Server (VPS).

Стосовно AJAX Systems, то дана компанія також має власні схожі мобільні додатки, так як, це є найкращим способом просування та підтримання продукту. Це не є додатковим бонусом, а необхідністю, яка зумовлена ринком.

1.4. Обґрунтування теми дипломного проекту

Давайте розглянемо окремі недоліки охоронних систем, які описані вище:

- Кастомізація мінімальна.

  Так як, такі системи являються монолітами із обмеженими можливостями зміни якихось конкретних модулів. Рішення, яке застосоване при розробці системи для даного дипломного проекту, може надати можливість використати будь-який технічний пристрій охоронного призначення (на технічних гаджетах, а саме мобільний телефон, планшет, ноутбук) з встановленим додатком та відкритим доступом до Інтернету;

- Складність монтажу та налаштування.

  Як правило, програмне забезпечення, яке встановлено та відповідні

правильні налаштування системи без професійного спеціаліста не обходиться, так як, він знає усі нюанси системи і зможе правильно все зібрати до кучі. Беручи додаток, то там більш простіше, необхідно виконати деякі прості кроки (дивитися рисунок 1.4.). Та при будь-якій зміні функціональності та місцеположення якогось із модулів все можливо виконати самостійно при малій затраті часу. І це надає високу гнучкість та занижує витрати на підтримку;

Рисунок 1.4.

- Зручність експлуатації.

  Бувають дуже часті випадки, коли користувач може забути вимкнути сигналізацію, тому спрацьовує хибний сигнал «Тривога» та може прибути охоронна служба. В цьому випадку, система має модель користувача, який дає можливість їй розпізнати, хто знаходиться на території об’єкту охорони.

В даному дипломному проекті буде використовуватися інтерактивна карта (доступ до Інтернет), яка може надати зручність та наглядність операторам з

різних охоронних служб.

2. АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ РОЗПОДІЛЕНИХ СИСТЕМ МОНІТОРИНГУ

2.1. Порівняльний аналіз програмних платформ для розробки серверних додатків.

На теперішній час можна назвати наступний найбільш популярний підхід для розробки орієнтованих в мережі Інтернет сервісів – підхід з використанням програмних засобів (LAMP): PHP, MySQL, Apache та Linux.

Будемо розглядати тільки обмеження веб-серверу Apache HTTP Server.

Слід вказати, що ядро Apache HTTP Server повністю побудовано на основі мови програмування C і реалізує одну із підсистем щодо контролю ресурсів та пам’яті, як називається пул (pool). Дана підсистема, в тому числі, контролює створення дочірніх процесів. При кожному наступному (новому) http-запиті, із pool витягується окремий процес (так як при інших випадках нам прийшлося б його створювати, то це додатковий час обробки запиту) для ізольованого контексту, і це дозволятиме паралельно та незалежно виконувати усі запити.

Слід назвати наступну проблему – використання ресурсів процесору та пам’яті, більша частина запиту клієнта – це очікування відповіді від бази даних, а також інших зовнішніх програмних інтерфейсів. Це являється неефективним, так як, кожен процес витрачає більшу частину ресурсів лише на очікування.

Цей підхід називається – синхронним вводом-виводом (див. рисунок 2.1.), тобто до моменту виконання операції система вводу-виводу просто блокується.

В протилежність існує підхід асинхронного вводу-виводу, який дозволяє

уникнути подібних проблем.

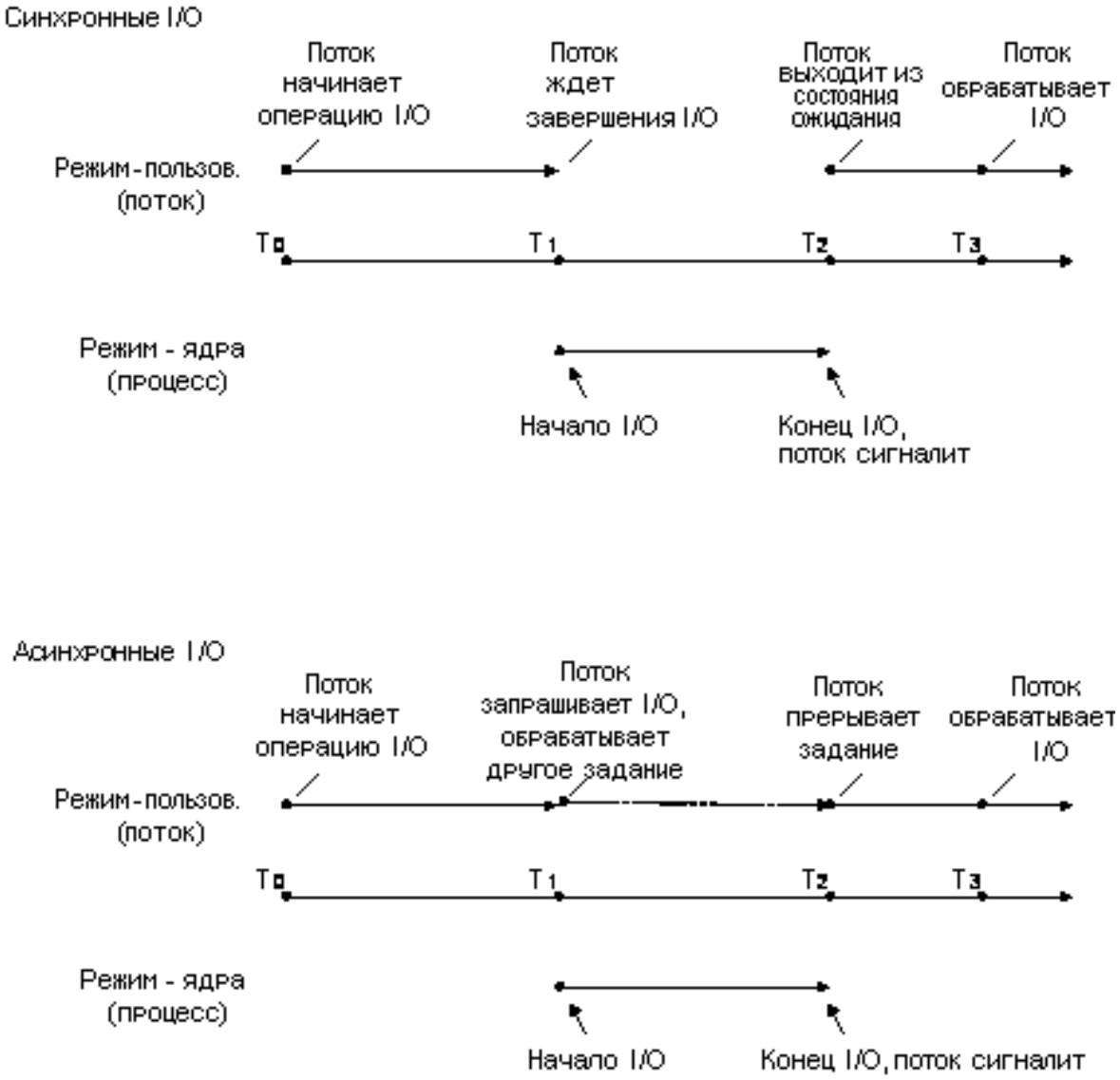

Рисунок 2.1.

NodeJS – являється на теперішній час найпопулярнішою платформою для веб-серверів із асинхронним вводом-виводом. Вона фокусується на асинхронному вводу-виводу, так як, побудована навколо бібліотеки libuv. Мовою програмування для NodeJS являється JavaScript, що робить її мовою загального значення, оскільки раніше ця мова використовувалася для взаємодії з документами в

браузерах. Також в цю платформу вбудований рушій від браузера Google Chrome V8 (дивитися рисунки 2.2., 2.3.).

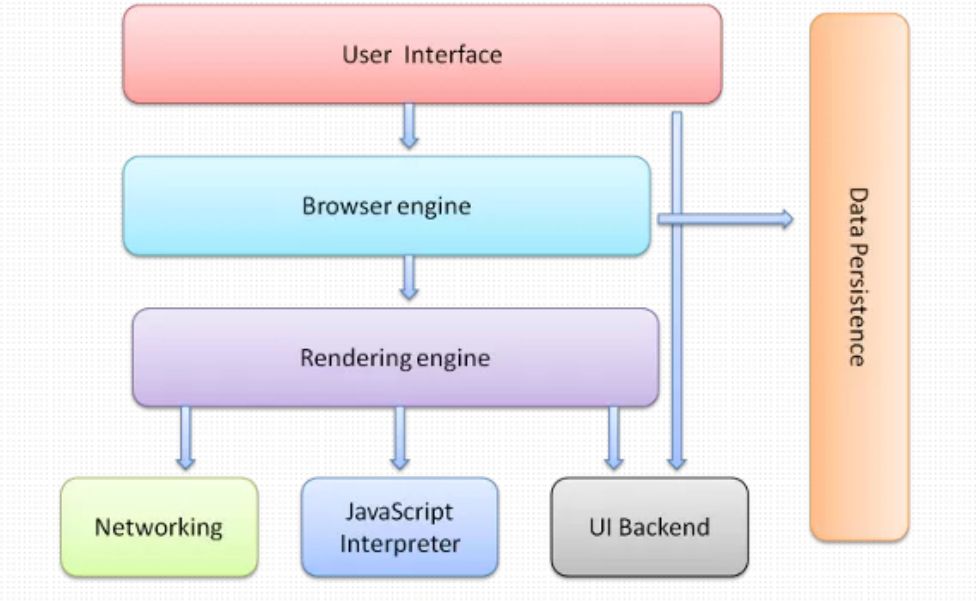

Рисунок 2.2.

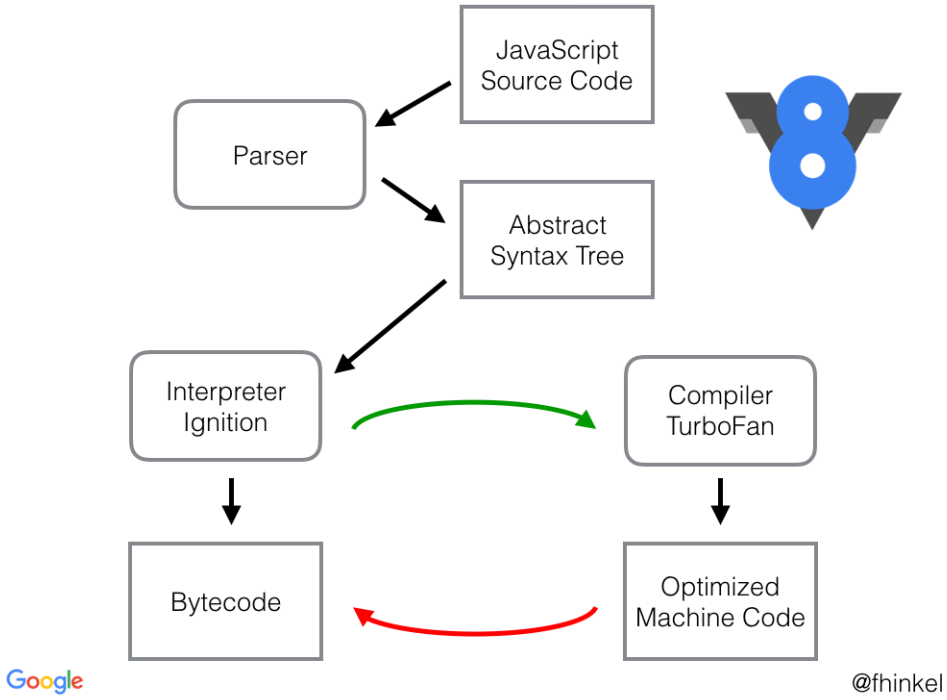

Рисунок 2.3.

Це все надало стрімкий розвиток цій мові програмування, дуже швидко вона змінюється, на даний час за її допомогою розробляються десктопні та вбудовані додатки.

Клієнтський код JavaScript виконується в одному потоці з асинхронним підходом. Здавалося б, як це сервер, який розрахований на велику кількість запитів в секунду, міг би працювати в одному потоці. Саме за допомогою libuv, вся робота з мережею, диском та іншими зовнішніми сервісами виконується належним чином. Вона написана на мові програмування C, що й реалізує паралельність виконання на нижньому рівні.

Цикл подій (Event Loop) надає можливість писати клієнтський код в асинхронному стилі. Через нього й відбувається вся комунікація між клієнтською програмою та системою вводу-виводу.

Отже, немає потрібності в створенні тисячі дочірніх процесів, які неефективно використовують ресурси процесора. Таким чином, можна сказати наступне, що всі запити обробляються в одному потоці, так як, процесорний час, який затрачався на очікування операції вводу-виводу, може використовуватися для обробки наступних запитів (дивитися рисунок 2.4.).

Відтепер створювання дочірніх процесів можна використати на інші цілі, а точніше для горизонтального масштабування. В цілому, є наступні види масштабування – горизонтальне та вертикальне.

При горизонтальному масштабуванні у функціонуючу систему можуть додаватися зовсім нові обчислювальні потужності, шляхом додавання в систему нових серверів або інших окремих вузлів (див. рисунок 2.5.). При такому, система буде працювати у виділеному кластері, в якому може знаходитися сотні

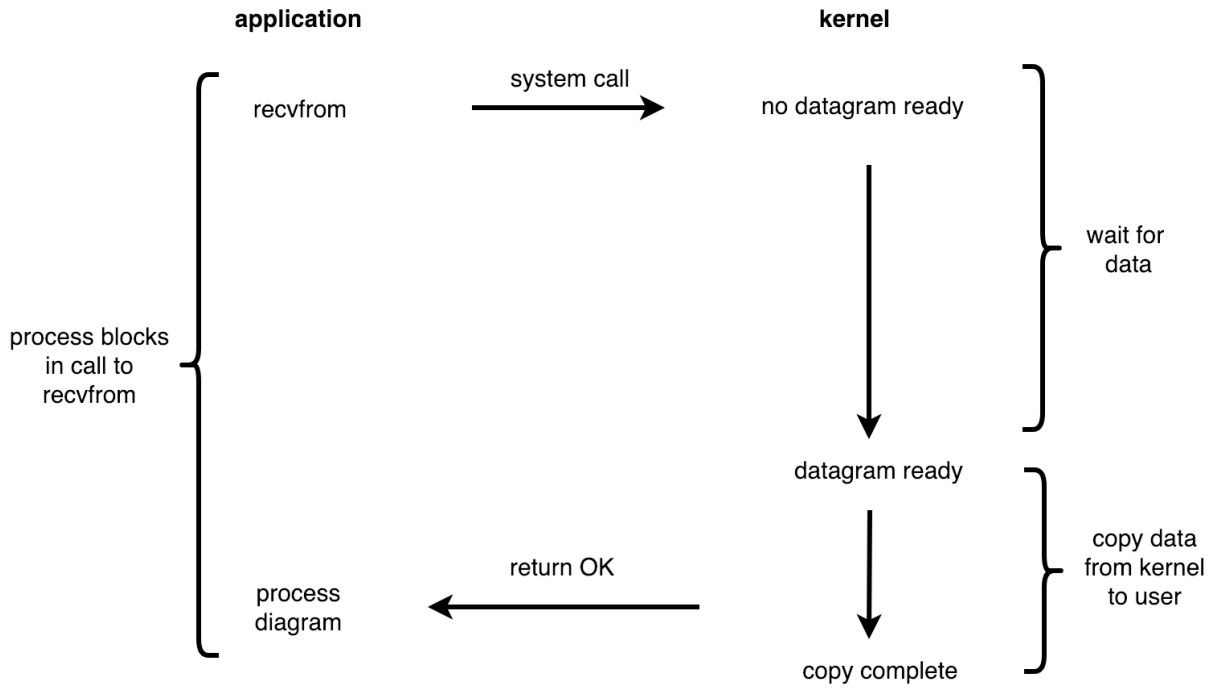

Рисунок 2.4.

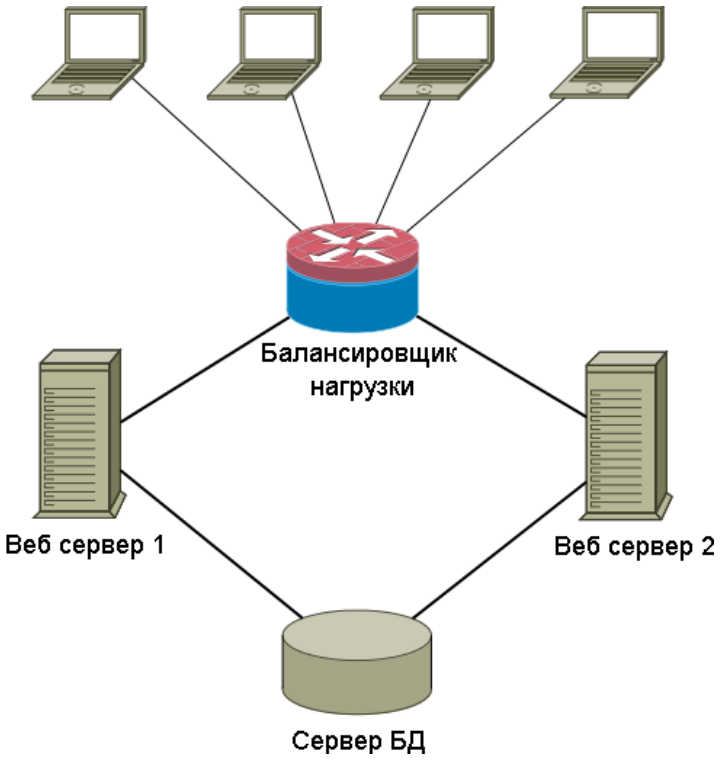

Рисунок 2.5.

обчислювальних вузлів. Головне, щоб це число можливо було змінювати в залежності від навантаження.

При вертикальному масштабуванні здійснюється покращення та модернізація обладнання – вдосконалення процесору та більше оперативної пам’яті (дивитися рисунок 2.6.). Таке масштабування можна назвати обмежувальним, оскільки вдосконалення обладнання має свою межу.

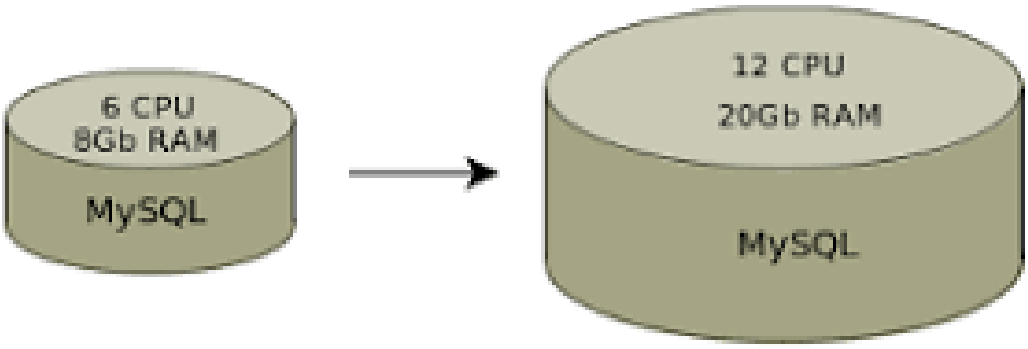

Рисунок 2.6.

В даному дипломному проекті буде використовуватися горизонтальне масштабування, котре реалізується на кількох рівнях роботи системи. Створення дочірніх систем – найнижчий рівень. Це є наступною причиною, тому, що цей дипломний проект буде використовувати для розробки програмну платформу NodeJS, оскільки вона з самого початку проектувалася для легкого горизонтального масштабування.

2.2. Порівняльний аналіз архітектурних стилів для проектування розподілених систем.

Будемо розглядати найбільш популярні способи комунікації вузлів розподіленої системи між собою:

- Message Queue;
- SOAP (Simple object Access Protocol);
- REST (Representational State Transfer).

Message Queue – це загальний підхід для передачі даних, за основу якого взято чергу повідомлень. Одними із найпопулярніших стандартів та реалізацій цього підходу слід вважати: RabbitMQ та Simple Queue Service (SQS) від Amazon Web Services (AWS). Цей варіант, мабуть, найбільше підходить для комунікацій мікро сервісів між собою та вирішує відповідні проблеми, але впровадження в систему його є найбільш складним, оскільки він потребує існування повідомлень в системі брокеру. Взявши RabbitMQ, то це являється додатковим сервісом, який потрібно підтримувати, а також, кожна черга, в залежності від мікросервісів потребує тонкого налаштування в частині мікросервісів, які нею користуються, та обраної стратегії обробки. Тобто, до кожної черги додається «мертва» черга, з метою зберігання неопрацьованих за певних причин повідомлень та послаблення настроювання всієї системи.

SOAP являє собою протокол, який певним чином працює поверх HTTP(S), хоча, слід зазначити, що він підтримує багато інших протоколів. Як недоліки, слід зазначити, що є неможливість роботи з іншими форматами даних, крім XML, який є надлишковим і має малу якість сприйняття. Швидкість передачі повідомлень стрімко падає через те, що SOAP додає власні дані.

REST – він не являється, а ні протоколом, а ні стандартом, оскільки це лише архітектурний стиль. Зазвичай, системи, які реалізовані із залученням REST, використовують такі стандарти: HTTP(S), JSON, URL.

Його комунікація систем має вигляд простоти та прямолінійності, але слід зазначити, що такі системи повинні відповідати наступним вимогам: відсутність стану додатка, модель клієнт-сервер, кешування. Даний архітектурний стиль викликає додаткові труднощі під час зростання навантаження на всю систему у цілому, це трапляється внаслідок використання мікросервісної архітектури. В даному дипломному проекті буде використаний даний варіант.

Давайте проаналізуємо монолітну та мікросервісну структуру між собою (дивитися рисунок 2.7.).

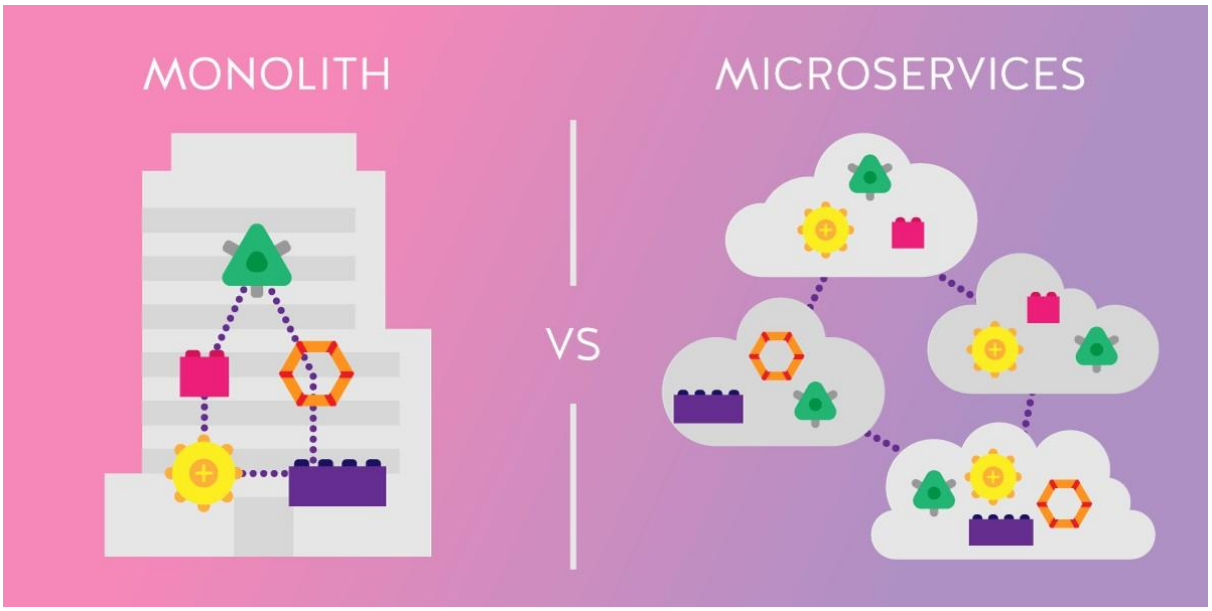

Рисунок 2.7.

Якщо взяти монолітну архітектуру, то вона передбачає реалізацію в рамках одного серверу всієї функціональності продукту. Через це може виникнути декілька проблем.

По-перше – під час використання концепції безперервної інтеграції та доставки продукту виникає значна складність розробки. Кожна функціональність, яка повинна бути доставлена користувачу потребує проходження багатьох етапів. Візьмемо за приклад наступне, з метою проходження автотестів необхідно розгорнути додаток в тестовому середовищі, а це в свою чергу тягне те, що необхідно його скомпілювати та зібрати, а у випадку моноліту, коли все це знаходиться в одній кодовій базі, це може відбуватися досить довго. Це стосується кожного етапу в цілому. Це дуже сповільнює швидкість розробки. В разі мікросервісного підходу, то кожна функціональність поділена безліччю сервісів, вони розгортаються, тестуються та доставляються незалежно одна від одної, і це набагато швидше.

По-друге – жодна з команд не несе відповідальності за частини продукту. При монолітній архітектурі весь проект знаходиться в єдиній кодовій базі, тому межі відповідальності розмиваються. Щодо мікросервісної архітектури поділ являється легшим, оскільки кожна команда відповідає за свій конкретний мікросервіс, отже й відповідальність стає чітко вираженою.

По-третє – свого роду проблема масштабування. В плані споживання ресурсів, окремий мікросервіс є передбачуваним і це дає можливість визначити приблизні необхідні апаратні характеристики. В залежності від навантаження такий мікросервіс легше масштабувати динамічно.

2.3. Порівняльний аналіз систем керування базами даних (СКБД)

Беручи даний дипломний проект, то можна сказати наступне, що роль ефективності DATABASE залежить від її можливості шардування (рисунок 2.8.).

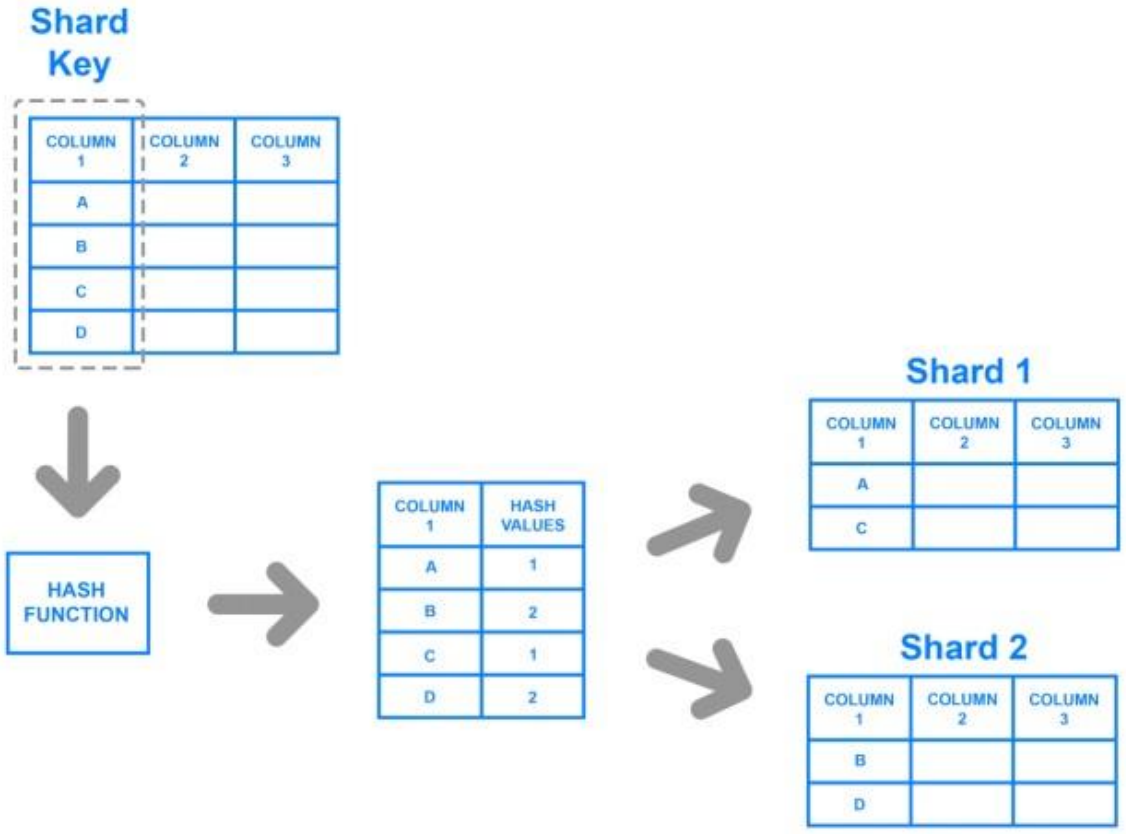

Рисунок 2.8.

Слід зазначити, що було вирішено використати систему керування базами даних (СКБД) MongoDB. Це NoSQL СКБД, які дає змогу налаштування шардування та реплікацію (див. рисунок 2.9.) декількома простими кроками, оскільки з початку вона була спроектована саме для цих цілей. Так як, в даній СКБД доволі легше оперувати записами в складній структурі, оскільки вони представлені у форматі JSON, і це дозволяє використовувати вкладеність. Оскільки NoSQL СКБД являється без схемним, то немає обмеження у

відповідності документів схемі. З помилковими документами, які записані в базі даних, це може додавати деякі проблеми, але цей підхід надає перевагу над класичними SQL СКБД навіть при великих об'ємах даних та складною структурою.

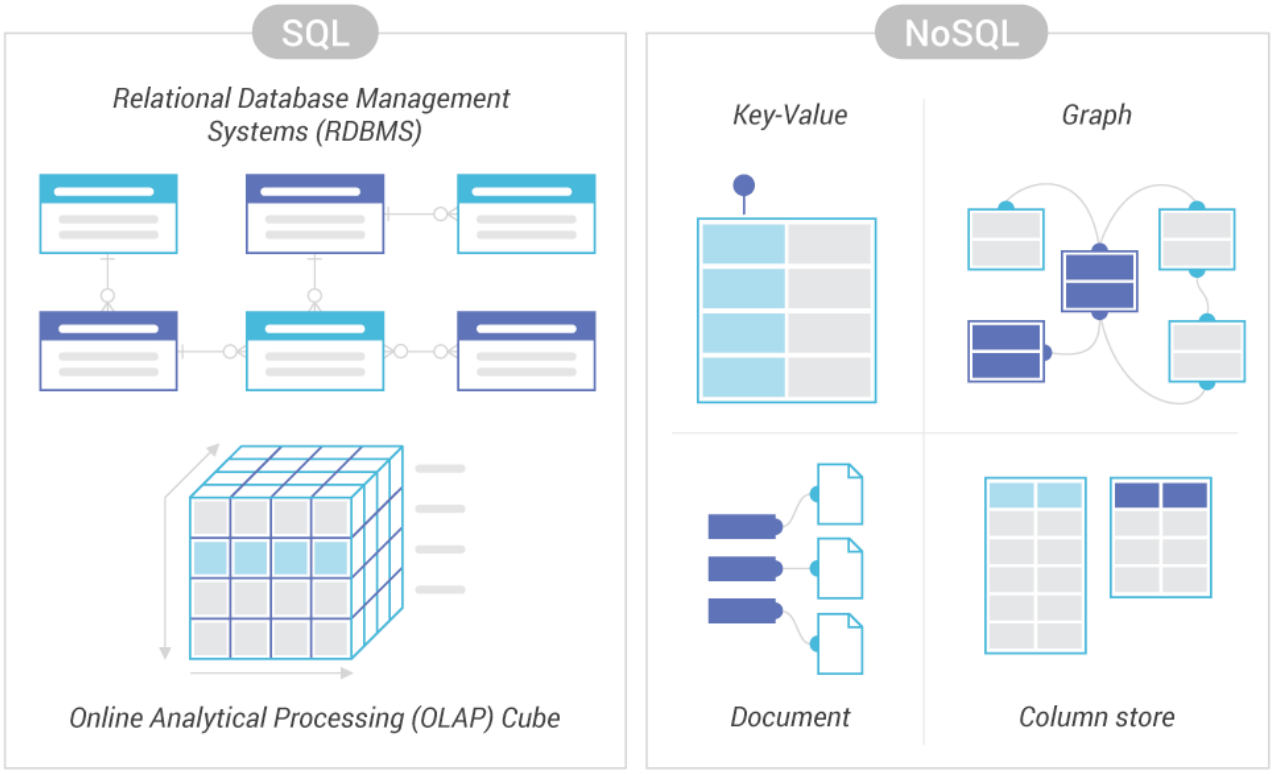

Рисунок 2.9.

2.4. Інші способи масштабування

Кешування – це інформаційна технологія, яка використовується для тимчасового зберігання зображень та веб-документів з метою зменшення серверних затримок.

Вона використовується для більшої ефективності системи.Content delivery network (CDN) є одним із способів. Мережа розповсюдження контенту – це мережа проксі-серверів та їх центрів обробки інформації, яка географічно розподілена.

Поширення сервісу просторово відносних кінцевих користувачів надає збільшення високої доступності та високої продуктивності. Більшу частину інтернет-контенту на сьогоднішній день обслуговує CDN. Сюди також слід віднести веб-об’єкти (графік, текст, сценарій), програми (портали, електронна комерція), завантажуванні об’єкти (мультимедійні файли, програмне забезпечення та документи), потокове мультимедіа, сайти соціальних медіа.

В екосистемі Інтернету, CDN являється шаром. Операторам CDN власники контенту (медіа компанії та постачальники електронної комерції) платять кошти за доставку їхнього контенту кінцевим користувачам. А CDN платять кошти операторам мереж та провайдерам за розміщення своїх серверів у їх центрах обробки даних.

CDN – це свого роду загальний термін, який включає різноманітні види послуг доставки контенту:

- Потокове відео;
- Прискорення веб- та мобільного контенту;
- Завантаження програмного забезпечення;
- Ліцензований та керований CDN;
- Балансування навантаження;
- Прозоре кешування та послуги для вимірювання продуктивності CDN;
- Комутації та аналітики.

Слід вказати, що вузли CDN розгортаються в декількох місцях зазвичай, дуже часто на декількох магістралях (дивитися рисунки 2.10., 2.11.).

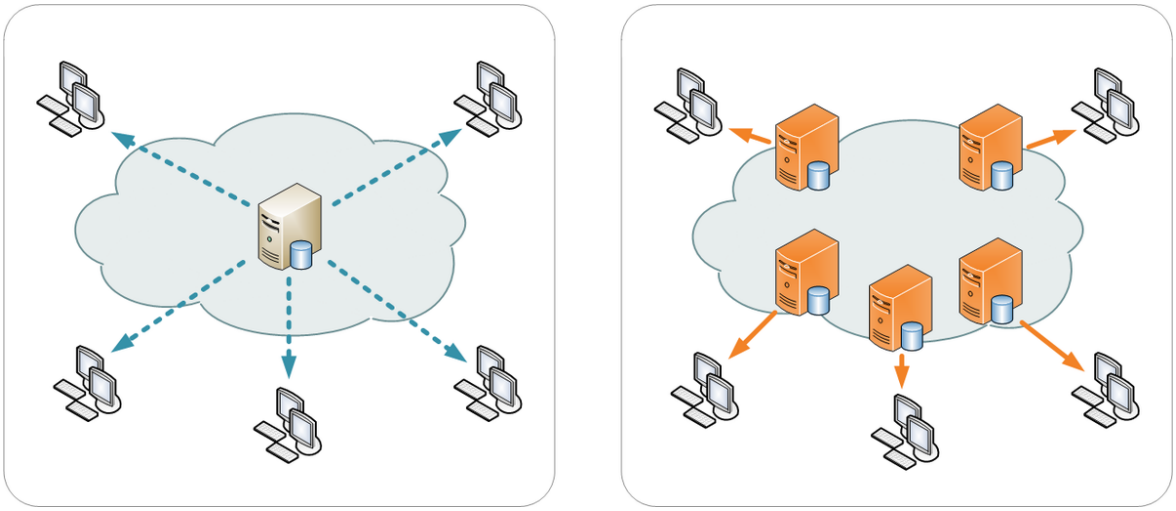

Рисунок 2.10.

Рисунок 2.11.

Залежно від архітектури кількість вузлів та серверів, які складають CDN можуть змінюватися. Деякі можуть досягати тисячі вузлів із десятками тисяч серверів на багатьох віддалених точках присутності. В цьому є свої переваги, а саме: підвищення часу завантаження сторінки, скорочення витрат на пропускну спроможність, збільшення доступності вмісту загалом.

Вузли є певним чином оптимальні і запити на вміст зазвичай алгоритмічно спрямовані на них. Під час самої оптимізації продуктивності має можливість вибирати місця, які найкраще б обслуговували користувачів. Щоб оптимізувати доставку через локальні мережі можна вибрати місця , які доступні користувачу із найменшою кількістю «хопів», найменшу кількість секунд у мережі від клієнта, або враховуючи інші фактори, то й найвищу доступність з точки зору продуктивності сервера.

PM2 Runtime – це інструмент, який дозволяє розгортати декілька екземплярів (скільки ядер на процесорі). Даний інструмент дозволяє зберігати програми, перезавантажуючи їх без простоїв та сприяти загальним завданням Devops.

Щоб покращити розподіл робочих навантажень на декількох обчислювальних ресурсах в обчислювальних системах слід використовувати балансування навантаження (дивитися рисунок 2.12.). Балансування навантаження – воно спрямоване на максимальну пропускну здатність, оптимізацію використання ресурсів, уникнення перевантаження будь-якого ресурсу та мінімізацію часу відгуку. За допомогою надмірності можна підвищити надійність і доступність. Балансування навантаження в свою чергу передбачає виділене програмне або апаратне забезпечення, сюди можна віднести багатошаровий комутатор чи серверний процес системи доменних імен.

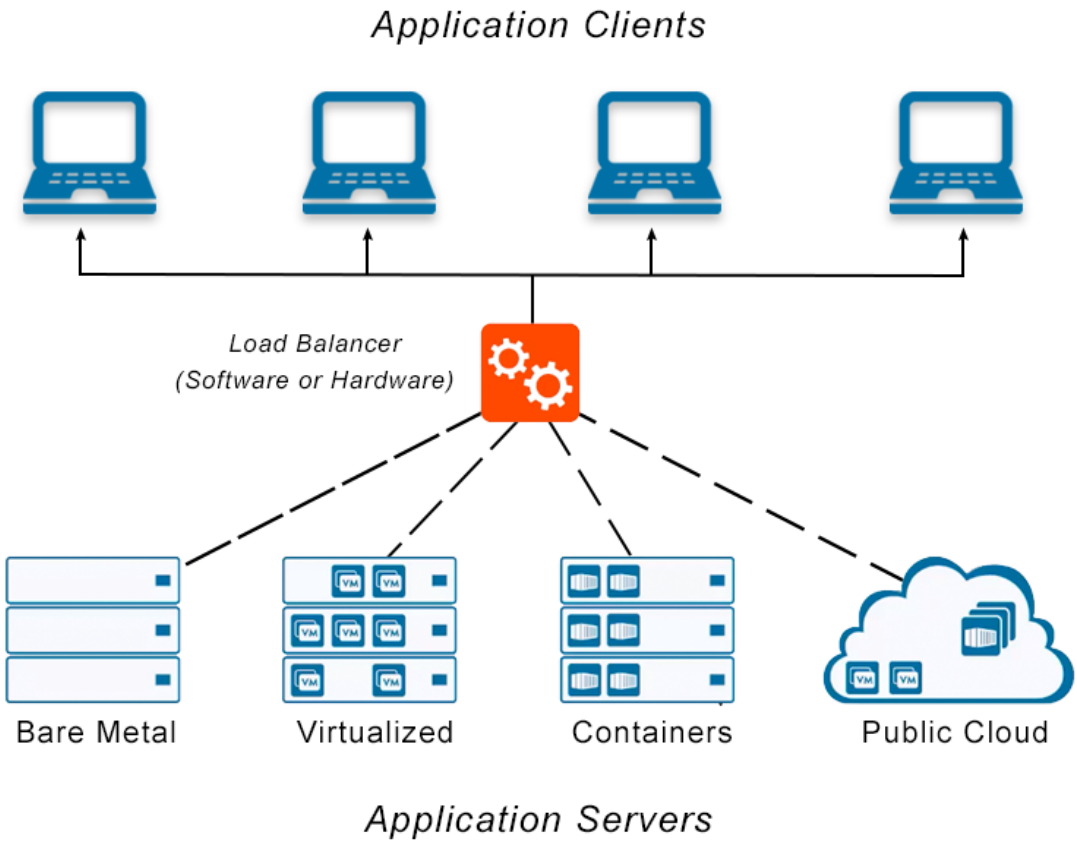

Рисунок 2.12.

Балансування навантаження розділяє трафік між мережевими інтерфейсами на основі мережевого сокету (це четвертий рівень моделі OSI), а зв'язок каналів – розподіл трафіку між фізичними інтерфейсами на нижньому рівні.

PM2 доволі корисний тим, що може додаток масштабувати по усіх доступних процесорах, при цьому створюючи кілька дочірніх процесів, які можуть використовувати один і той же порт сервера. Для протоколів HTTP, TCP, UDP він чудово працює і дає можливість збільшити продуктивність на порядок, на 16 основних машинах. Слід зазначити, що він зменшує простої на оновленнях.

Цей інструмент є доволі потужним для розгортання систем з уже вбудованим масштабуванням та балансувальником.

3. Структура системи та опис роботи модулів

3.1. Система керування базою даних (СКБД)

Слід вказати, що база даних складається з наступних колекцій: USERS та SECURITY KEY.

Users – в даній колекції зберігається уся інформація про користувача (його унікальне ім’я та дані у наступному форматі base64 (дивитися рисунок 3.1.).

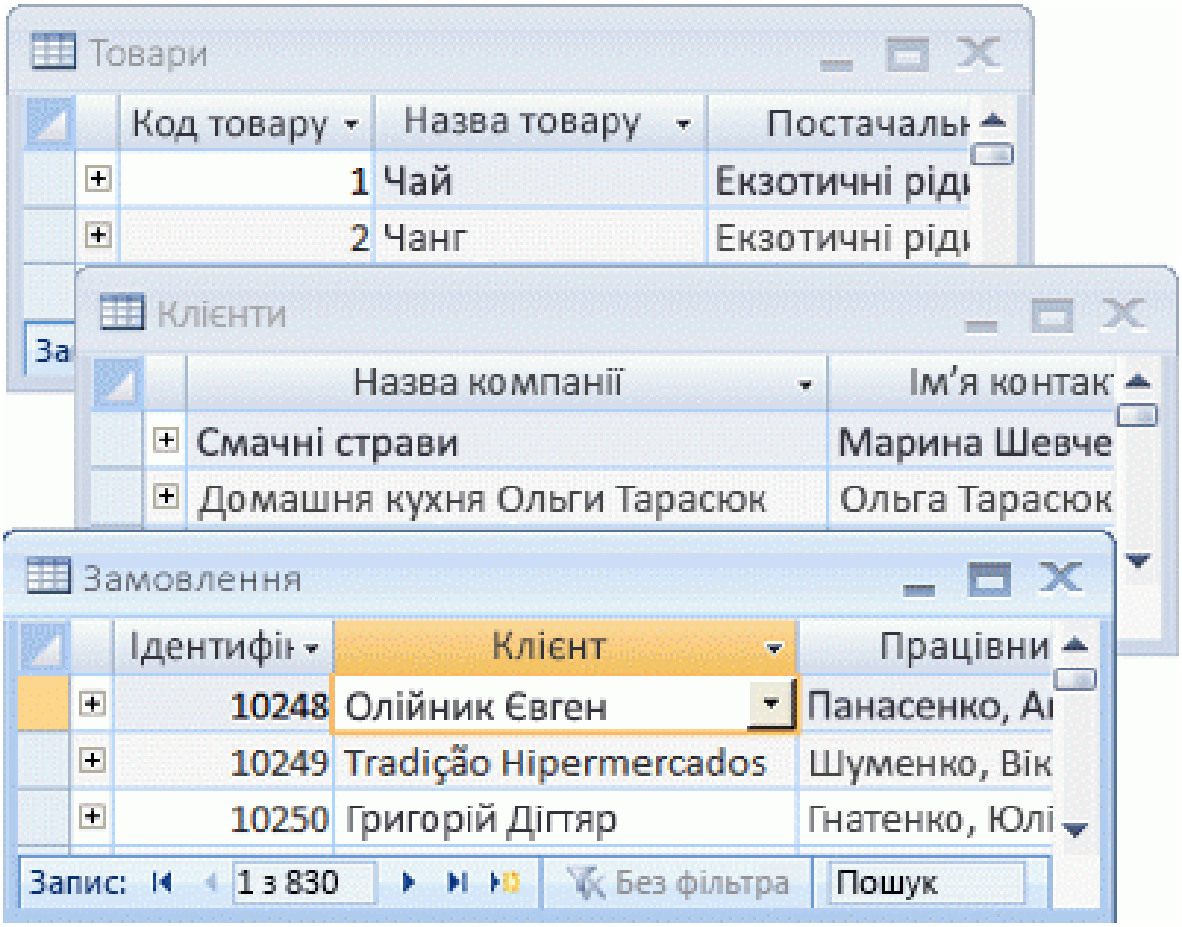

Рисунок 3.1.

Розширені данні пропускного режиму, ідентифікатори перепусток, рівня доступу на певні об’єкти та їх приміщення знаходяться в іншій колекції – security key (дивитися рисунок 3.2.). Для правильної роботи підсистемі потрібна відразу ж вся інформація про користувачів після їхньої класифікації, отже всі ці дані зберігаються в одному документі. Розміри даного документу, очевидно ж будуть різко збільшуватися, оскільки дані про ідентифікатори (перепустки) займають чимало місця. В СКБД MongoDB за замовчуванням обмеження на розмір одного документу встановлено у розмірі 16 мегабайт. Для того, щоб обійти ці обмеження, MongoBD GridFS-інструмент.

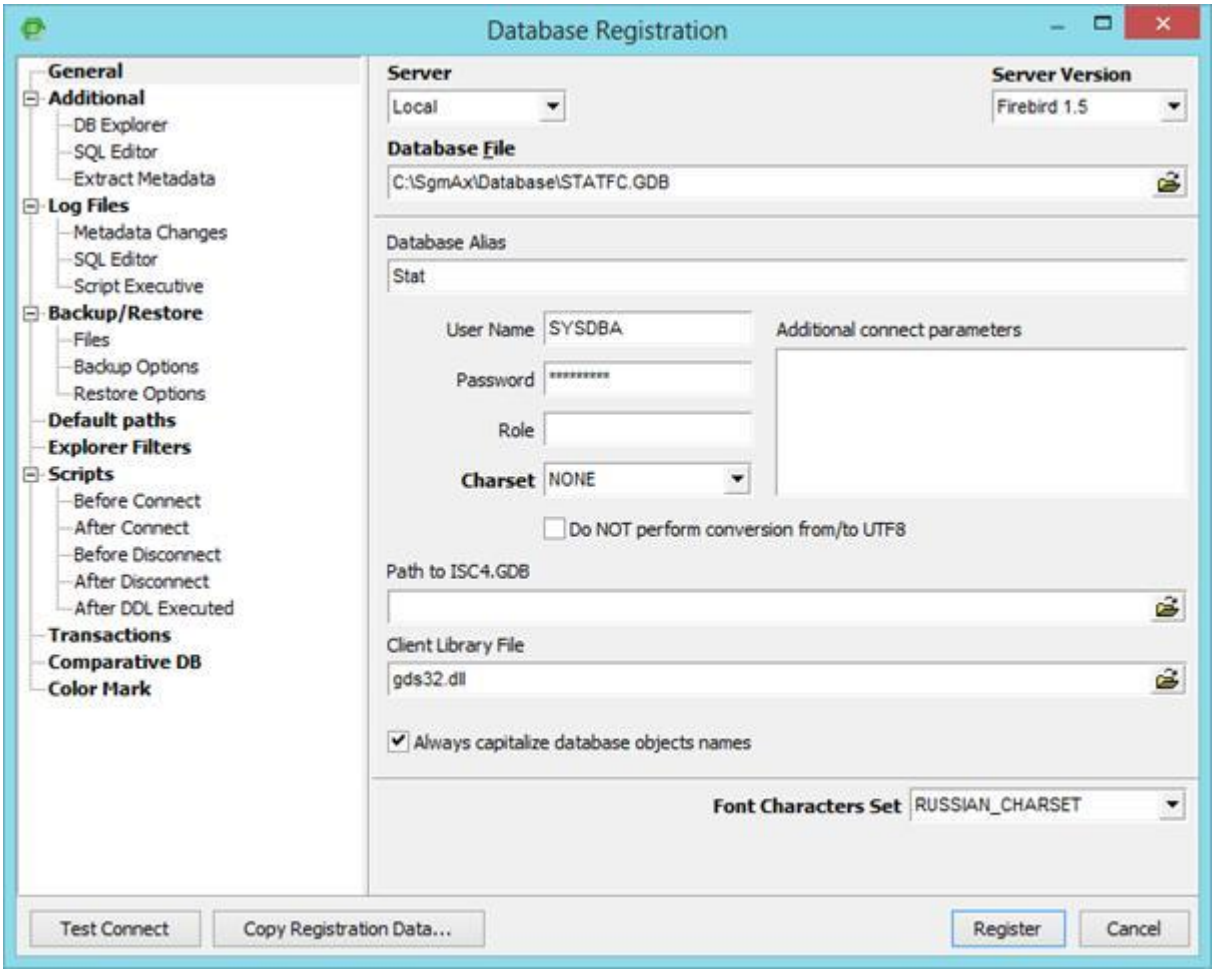

Рисунок 3.2.

GridFS здійснює ділення файлу на частини чи фрагменти замість того, щоб зберегти файл в одному документі, зберігає кожен фрагмент, як окремий документ. За замовчування 255 Кб – стільки GridFS використовує розмір фрагменту. За винятком останнього фрагменту, GridFS ділить файл на блоки розміром 255 Кб. Останній файл буде мати точно необхідний розмір. Також, в свою чергу, файли, котрі не перевищують розмір фрагменту, будуть використовувати стільки ж місця, стільке це необхідно.

Слід вказати, що GridFS буде використовувати лише дві колекції для зберігання файлів. Одна з цих колекцій буде зберігати файлові шматки, а решта файлів будуть збережені в метадані. Кожна колекція в GridFS буде детально описана у розділі Колекція.

Якщо є необхідність, то драйвер буде автоматично збирати цей файл з фрагментів, варто лише надіслати запит до GridFS. GridFS дає можливість здійснювати також діапазонні запити на файли. Можливо також отримати доступ до даних із довільних розділів файлів.

GridFS корисний також для зберігання різних файлів, при яких необхідно мати доступ, без потрібності завантаження всього файлу в пам’ять, а не тільки для зберігання файлів, які перевищують 16 Мб.

В окремих випадках, збереження великих файлів може бути більше ефективним в базі даних MongoDB, чим у файловій на рівні системи.

1) Для зберігання необхідної кількості файлів, використовуйте GridFS, це якщо файлова система ставить обмеження кількості файлів в каталозі.

2) Для того, щоб відтворювати розділи файлів, при цьому, не читаючи весь файл в пам’ять, краще використовувати GridFS, це якщо мала місце

необхідність отримання доступу до інформації з частин великих файлів.

3) У разі того, якщо необхідно, щоб метадані та файли автоматично розгорталися та синхронізувалися в ряд засобів та систем, також модна використати GridFS. MongoDB, використавши географічно розподілений набір реплік, може автоматично поширити файли та їхні метадані на декілька екземплярів mongod.

Якщо є необхідність оновити вміст усього файлу атомічно, GridFS краще не використовувати. Як альтернативу, можна зробити наступне: зберегти кілька версій кожного файлу та вказати в метаданих поточну версію файлу. Завантаживши нову версію файлів, то є можливим оновити поле метаданих, яке вказуватиме на «останній» статус, після чого можна видаляти попередні версії.

Щоб не використовувати GridFS, можна зберігати кожний файл в одному документі, це якщо файли менші, чим обмеження розміру документу у BSON. Для зберігання двійкових даних можна використати BinData.

Як вказано на рисунку 3.3. GridFS буде зберігати файли в двох колекціях:

1) Колекція chunks буде зберігати двійкові фрагменти даних. Слід зазначити, що кожен документ в цій колекції являється окремим фрагментом файлу, вираженим у GridFS;

2) Метадані файлів – зберігає колекція files. Також, кожний документ в цій колекції являється файлом в GridFS.

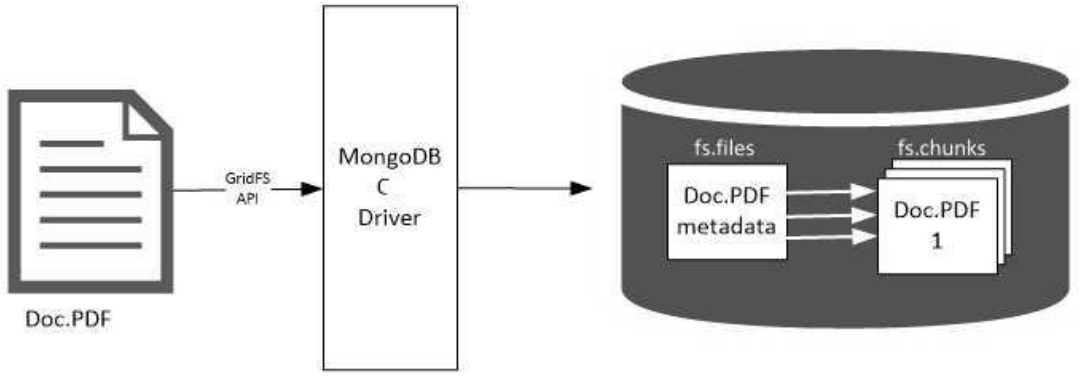

Рисунок 3.3.

3.2. Модуль обробки технічних сповіщувачів

Модуль приймає запити на наступні ресурси:

1. GET/info;
2. POST/user;
3. POST/security key.

Ідемпотентність – це властивість бінарних, а також унарних операцій в логіці та алгебрі.

Важливою частиною є те, що ідемпотентними мають бути всі операції системи, оскільки доволі часто проблеми, які виникають внаслідок мережі, запити можуть повторюватися, то цей підхід буде найважливішим у проектуванні та розроблювані програмного забезпечення (ПЗ), тим паче, з мікросервісною архітектурою, в тому випадку, коли сервіси комунікують з одним користувачем. Слід назвати один із прикладів проблеми, яка може виникнути, наведено у рисунку 3.4. Цей випадок показує те, що запит може, все-таки, бути прийнятим та обробленим системою, але, в свою чергу, відправник не буде бачити, що запит відправлений, отже він буде знову направлений.

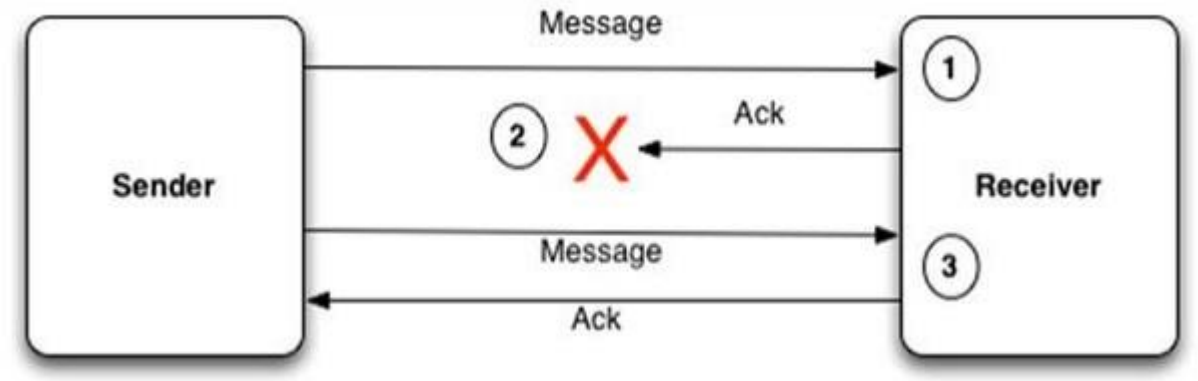

Рисунок 3.4.

Слід зауважити, що метод HTTP стає ідемпотентним, в разі, коли аналогічний запит, який зроблено один чи декілька разів поспіль, має такий же ефект, при цьому не змінюючи стану системи. Тому ідемпотентний метод не матиме побічних ефектів, окрім збору статистичних даних чи подібних операцій.

Запит GET/info буде повертати інформацію на цей самий момент про систему, а саме: кодову назву, версію та інше. Версія системи призначена для того, щоб мати визначення того, яка із функціональності вже в наявності та якої поведінки слід чекати від системи, оскільки при кожній версії вона буде мінятися. Своє власне версіонування буде використовуватися й для кожного окремого мікросервісу. Саме для цього, буде забезпечуватися метод під назвою Semver.

«Пекло залежностей», саме це матиме важливе місце у розробці програмного забезпечення. З кожним разом, як система буде зростати та чим більше пакетів буде інтегруватися в програмне забезпечення, тому буде набагато більше шансів попасти в це місце.

Слід зазначити, що чим більше випускається нових версій пакетів, а тим паче, у системах з багатьма залежностями, це все може швидко стати кошмаром.

Можна опинитися у небезпеці блокування версії, це якщо специфікації залежностей занадто тісні (унеможливлення оновлення пакету без випуску новітніх версій кожного із залежних пакетів).

Для вирішення проблеми, яка виникла, було запропоновано набір вимог та правил, які вказують, коли номери версій збільшуються та призначаються. Дані правила можуть ґрунтуватися (в залежності від потреби) на практиках використання, що в закритому, що у відкритому програмному забезпечені, які існуючі та досить поширені на передодні. Для того, щоб ця система запрацювала, спершу оголосимо відкритий API. Обов'язковою умовою є те, щоб ця API була чіткою та точною. Після проведеної роботи, API загальнодоступний став проідентифікованим, необхідно оголосити про зміни із збільшеннями до номера нової версії. Буде розглянуто формат XYZ (Major.Minor.Patch). Помилки, які не впливають на роботу API та будуть виправлені, тягне за собою збільшення Patch версії, збільшення функціональності чи зворотні сумісні зміни API збільшать Minor версію, а несумісні – збільшать Minor версію.

Це не вважається новітньою чи революційною ідеєю. Однак, без належної відповідності певним формальностям специфікацій номерів версій, безрезультатні для управління залежностями. Надавши ім'я та чітке визначення, має досить легке місце подачі інформації користувачу вашого програмного забезпечення про свої міри. Після того, як користувач зрозуміє, що вимагається, може бути дуже легко розроблено гнучкі специфікації залежності.

Запит POST/user забезпечує створення нового користувача, в основі чого лежить його ім'я та відповідні дані. При цьому буде відбуватися аналіз сповіщення, суб'єкта, який проник на об'єкт охорони за допомогою датчиків руху для здійснених фотографій із подальшою побудовою унікальної моделі опорно -

векторної машини (ОВМ) суб'єкта – близько 9-ти 128-ми вимірних векторів на основі п'яти фотографій та замірів датчиків руху. Дев'ять векторів значно підвищують відсоток достовірних розпізнавань, а ніж один. З метою побудови такої моделі буде використано згорткову нейронну мережу (ЗНМ).

Алгоритмом опорно-векторної машини буде пошук гіперплощини N мірному просторі (N – це кількість ознак), яка буде чітко класифікувати точки даних [6].

Для відокремлення двох класів точок даних, ми можемо вибрати велику кількість можливих гіперплощин. Метою даного прикладу буде наступне: пошук площини, котра має максимальний запас, а точніше максимальну відстань, яка буде між точками даних обох класів (дивитися рисунок 3.5.).

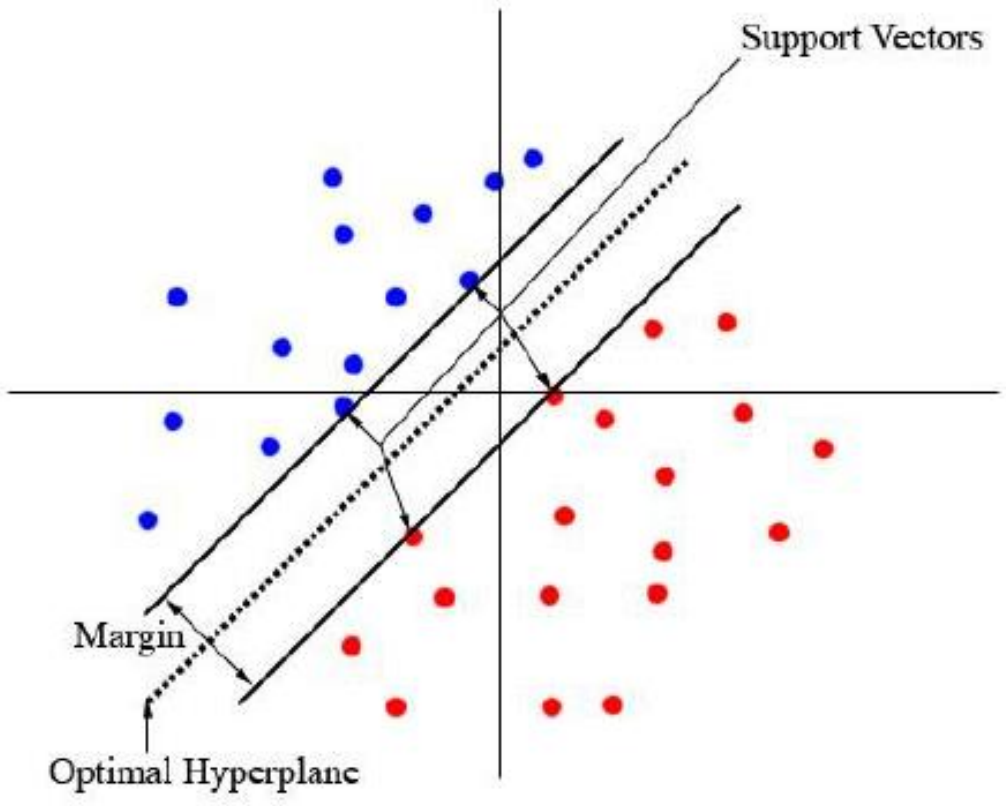

Рисунок 3.5.

Для класифікації точки даних, допоможуть межі рішень, які мають назву гіперплощини. Якщо точки даних, які падають на будь-яку зі сторін гіперплощини, то їх можливо віднести до різних класів. Також слід зазначити, що розмір гіперплощини буде залежати від кількості функцій. Якщо, наприклад, кількість вхідних функцій матиме значення 2, то слід вважати, що гіперплощина являє собою лише лінію. Якщо ж число вхідних ознак матиме значення 3, то тоді, гіперплощина являтиме собою двовимірну площину. Навіть важко представити, що буде, коли кількість ознак становитиме понад 3 (див. рисунки 3.6. та 3.7.).

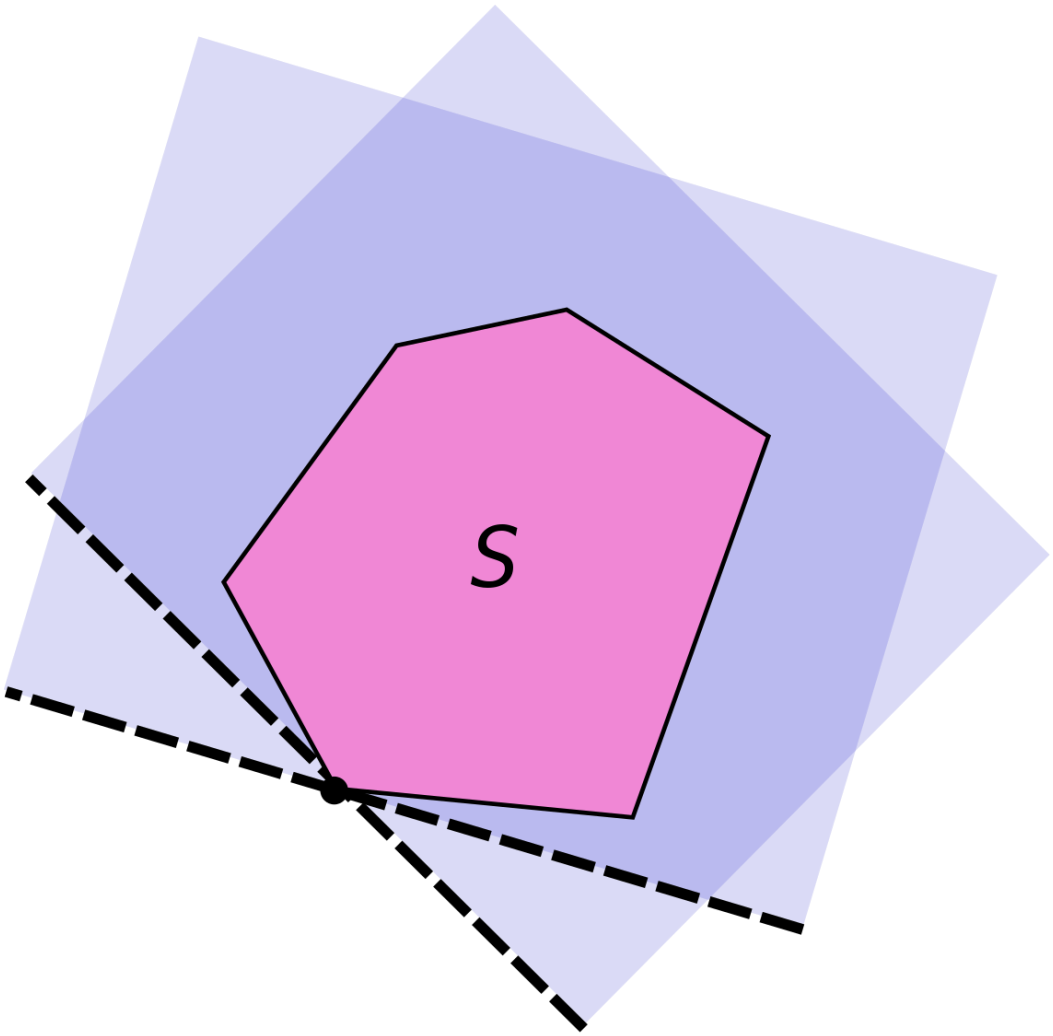

Рисунок 3.6.

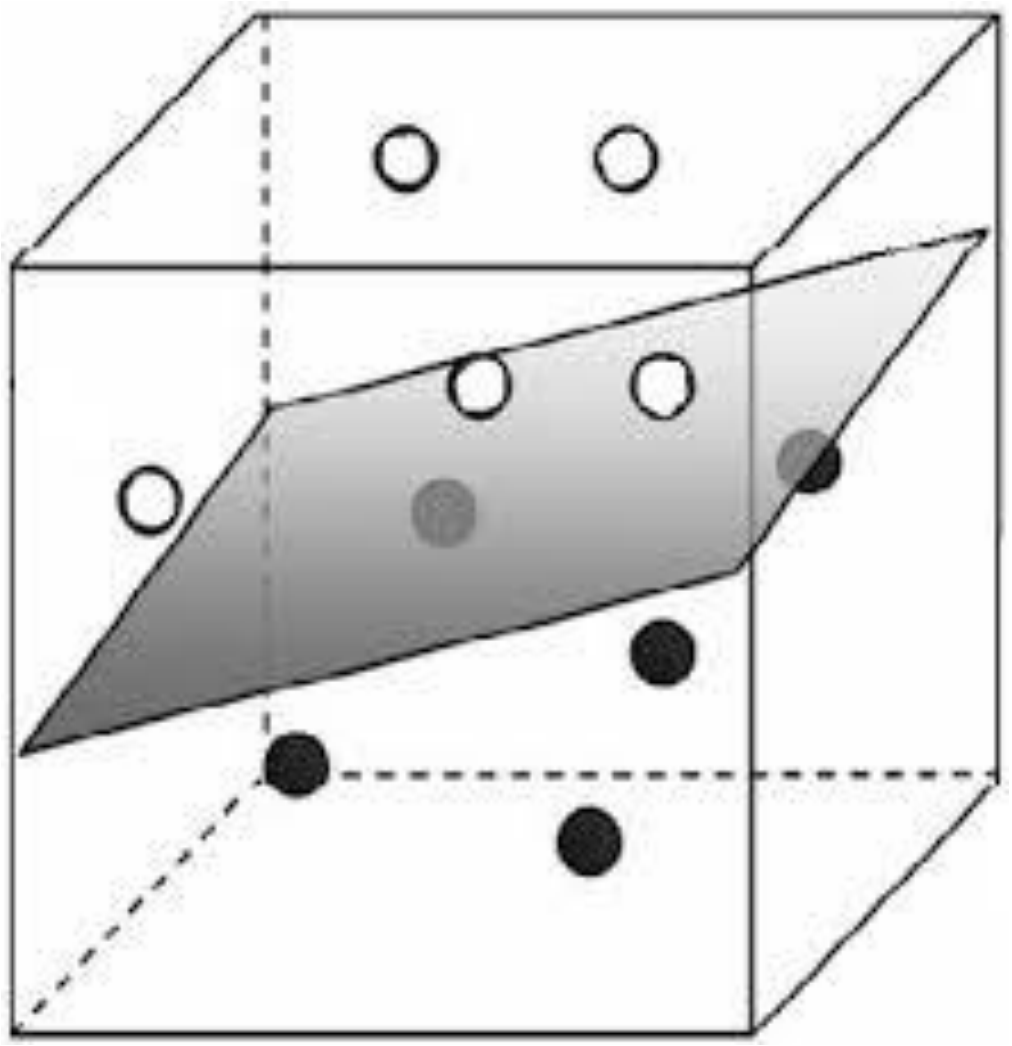

Рисунок 3.7.

Точки даних являють собою векторами підтримки, котрі мають найближче місце до гіперплощини та вносять корективи на положення і орієнтацію гіперплощини (дивитися рисунок 3.8.). З метою максимізації відстані для класифікатора, забезпечують використання цих векторів підтримки. Та в свою чергу, усунення векторів підтримки змінить саме положення гіперплощини. Ці пункти допоможуть побудувати ОВМ.

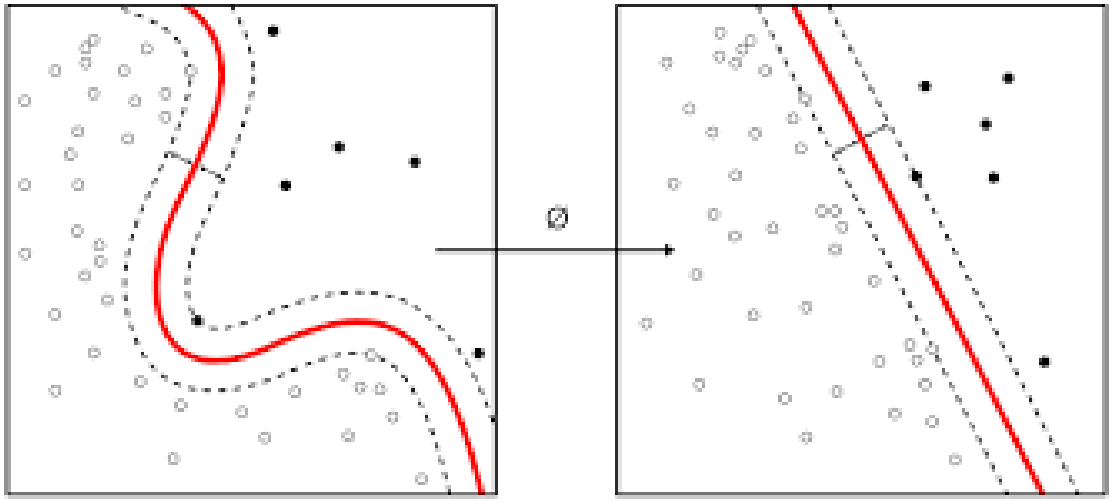

Рисунок 3.8.

Максимізація відстані між точками гіперплощини та даних є основною метою алгоритму ОВМ. Завісні втрати – це функція втрат, яка допоможе максимізувати відстань (дивитися рисунок 3.9.).

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Рисунок 3.9.

Якщо фактичне та передбачене значення матимуть один і той же знак, то вартість становитиме 0. При умові, якщо це не відбувається, тоді здійснюється обчислення значення втрат. А також, в свою чергу, додається параметр регуляризації функції втрат. Збалансування максимальної відстані та втрати і є основною метою параметра регуляризації. Функції втрат виглядатимуть після додавання параметра регуляризації наступник чином (дивитися рисунок 3.10.).

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Рисунок 3.10.

Після того, як отримали функцію втрат, забезпечується використання частково похідних відносно ваг, для того щоб знайти градієнти. Також, за допомогою них ми можемо оновити ваги (дивитися рисунок 3.11.).

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} \left(1 - y_i \langle x_i, w \rangle\right)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Рисунок 3.11.

Коли модель правильно передбачає клас точки даних, тобто відсутня неправильна класифікація, тоді потрібно оновити лише градієнт від параметра регуляризації (дивитися рисунок 3.12.).

$$w = w - \alpha \cdot (2\lambda w)$$

Рисунок 3.12.

Коли виявлено помилку класифікації, а саме, модель на прогнозуванні класу точки даних помиляється, втрати разом з параметром регуляризації включаються для виконання оновлення градієнта (дивитися рисунок 3.13.).

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

Рисунок 3.13.

ЗНМ являє собою алгоритм глибокого навчання, він може отримувати вхідне зображення (сигнал), назначати важливість (упередження та навчальні ваги) різним об’єктам та аспектам на відображені, а також вміти диференціювати один з іншого (дивитися рисунок 3.14.).

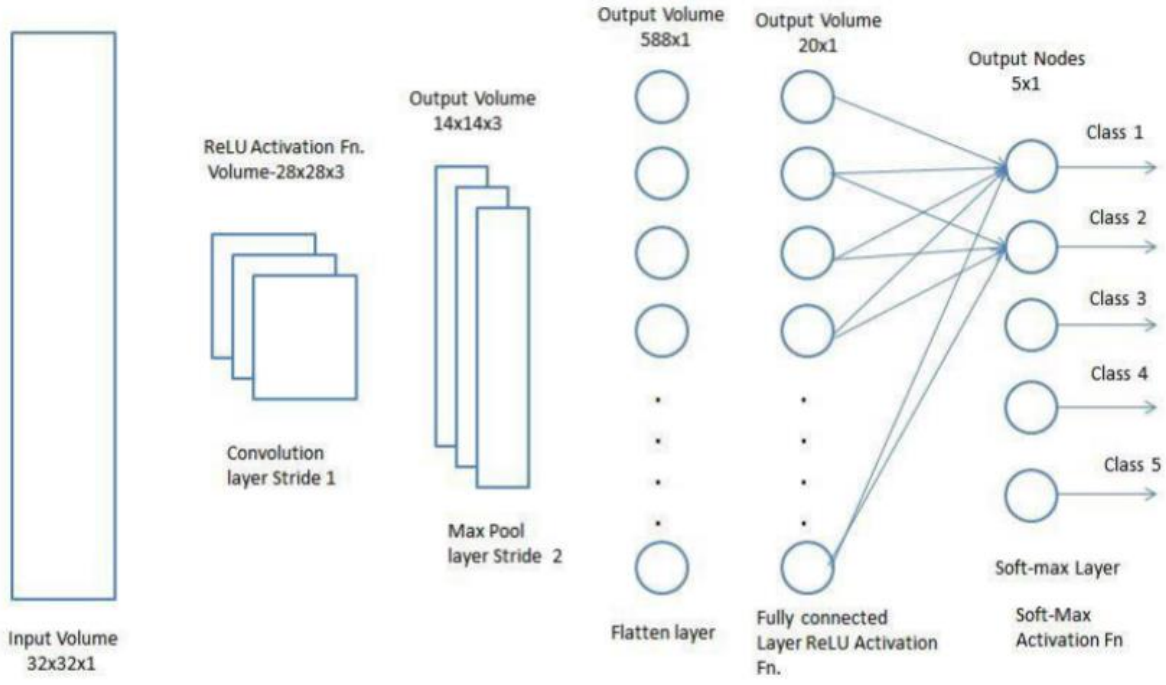

Рисунок 3.14.

В порівнянні з іншими алгоритмами класифікації, в ЗНМ попередня обробка, яка необхідна значно менша. ЗНМ мають можливість обробляти ці фільтри та характеристики, хоч в примітивних методах фільтри сконструйовані вручну із достатньою кількістю тренувань.

Аналогом архітектури ЗНМ можна назвати архітектуру зв'язності нейронів у людському мозку. Рецептиве поле – це свого роду окремі нейрони, які реагують на подразники, тільки в обмеженому регіоні поля зору. З метою охоплення всієї зони зору, здійснюється перекриття колекції таких полів.

ЗНМ має можливість позитивно отримувати тимчасові та просторові залежності у зображенні та відповідних датчиках руху за допомогою відповідних фільтрів. Набір зображень за рахунок зменшення кількості задіяних параметрів та повторного використання ваг, тому ця архітектура краще підходить. Тому, дану мережу ми можемо навчити краще розуміти складність зображення да різноманіття сигналів, котрі надійшли з певних датчиків.

Тому, в цьому дипломному проекті, векторне вбудовування – це показ зображення та відповідних сигналів у числові векторні представлення їх, шляхом проходження через згортові нейронні мережі.

Векторну відстань ми можемо використати для розрахування подібності між двома векторами, так як, дані векторні вбудування показані у спільному векторному просторі. В контексті розпізнавання суб'єкта, дана векторна відстань застосовується з розрахунку – наскільки подібні декілька суб'єктів. Окрім цього, дані вбудування можуть використовуватися вхідними елементами в класифікації, регресії чи кластеризації.

Для розроблення використовується бібліотека OpenCV, хоча вона і

реалізована за допомогою C#, але за допомогою аддонів у Node.js всякі коди на C та C++ можуть бути підготовлені з метою використання у Node.js. – JavaScript являється мовою програмування для даної програмної платформи.

Слід вказати, що адонном Node.js являються динамічна зв’язність спільних об’єктів, що написані мовою програмування C++, котрі ми можемо завантажити до Node.js, при цьому використавши функцію require(), при тому, що якби вони були звичайним модулем, їх також можна використовувати.

Крім того, потрібно вказати наступне, що на POST/security_key запити обробляються, при цьому аналізуються знімки, сигнали з датчиків руху з безпосереднього об’єкту охорони і намагається розпізнати дані (фото обличчя, висота та ширина суб’єкту) та визначає хто ж саме знаходиться на об’єкті. Та в свою чергу, оброблюються дані про місцерозташування датчиків руху, місцерозташування камер та самого власника. При позитивному оброблені даних, вони направляються на інші мікросервіси, котрі відповідають за сповіщання.

3.3. Модуль сповіщення.

Стосовно модулів сповіщення, то вони являють собою окремі мікросервіси, які отримують запити від модулів обробки даних на ресурс POST/alarm з повністю необхідною інформацією. За допомогою протоколу WebSocket всі клієнти цієї системи підключаються до вище зазначеного модуля. У разі прийняття даного запиту, інформація про перебування суб’єкту на об’єкті відразу ж направляється всім клієнтам системи (дивитися рисунок 3.15.).

WebSocket являє собою комп’ютерний комунікаційний протокол, котрий надає повний дуплексний канал зв’язку на одне з’єднання TCP. Він різниться з HTTP. Що один, що інший протокол мають рівень 7 у моделі OSI та залежать від

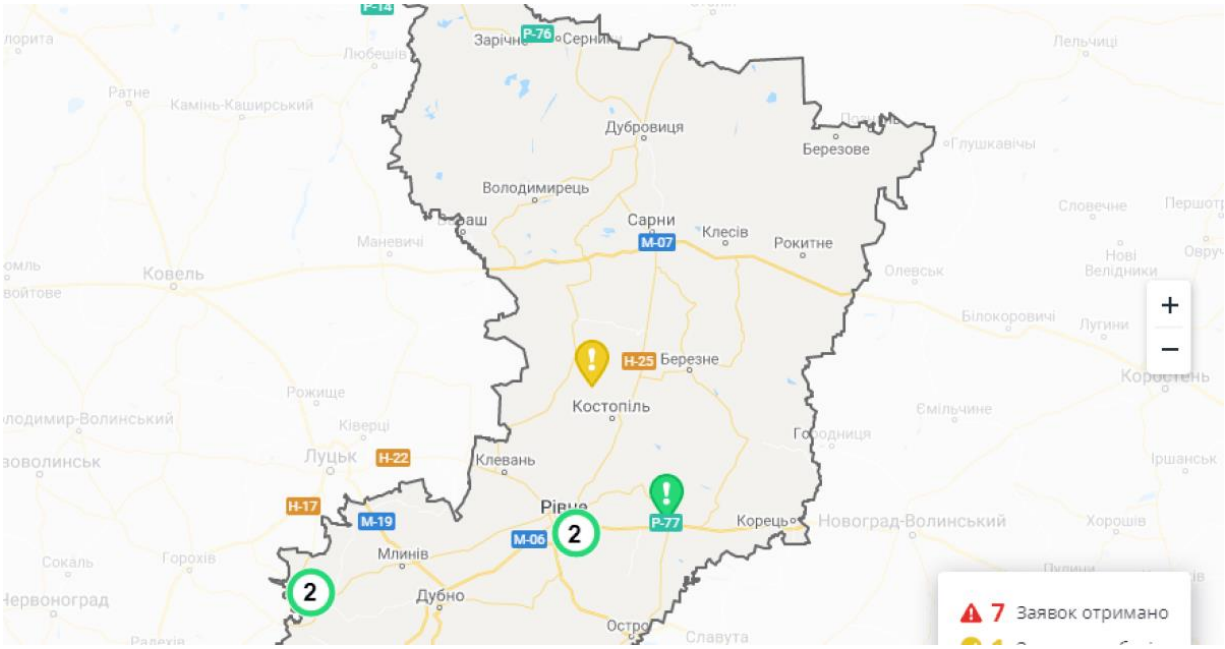

Рисунок 3.15.

TCP на 4 рівні. Хоч вони і різняться, RFC (Request for Comments) 6455 стверджує – WebSocket назначений з метою роботи над HTTP портами 80 та 443, та для підтримки HTTP посередників та проксі. І цей момент, робить його сумісним з протоколом HTTP. З метою переходу від протоколу HTTP до WebSocket використовується заголовок HTTP Upgrade, за рахунок цього досягається сумісність WebSocket.

Протокол WebSocket дає можливість з меншими накладними витратами взаємодіяти між веб-сервером та веб-браузером (рисунок 3.16.). За допомогою стандартизованого способу передачі контенту від сервера до клієнта, без його попереднього запиту, що дозволяє сповіщенням передаватися вперед та назад, при цьому, зберігаючи відкрите з'єднання. Тому відбувається двостороння комунікація між сервером та клієнтом. Корисним для середовищ є те, що комунікація здійснюється через TCP-порт 80 (чи 443 при TLS зашифрованих з'єднань), так як, блокують не веб-підключення до Інтернету за допомогою

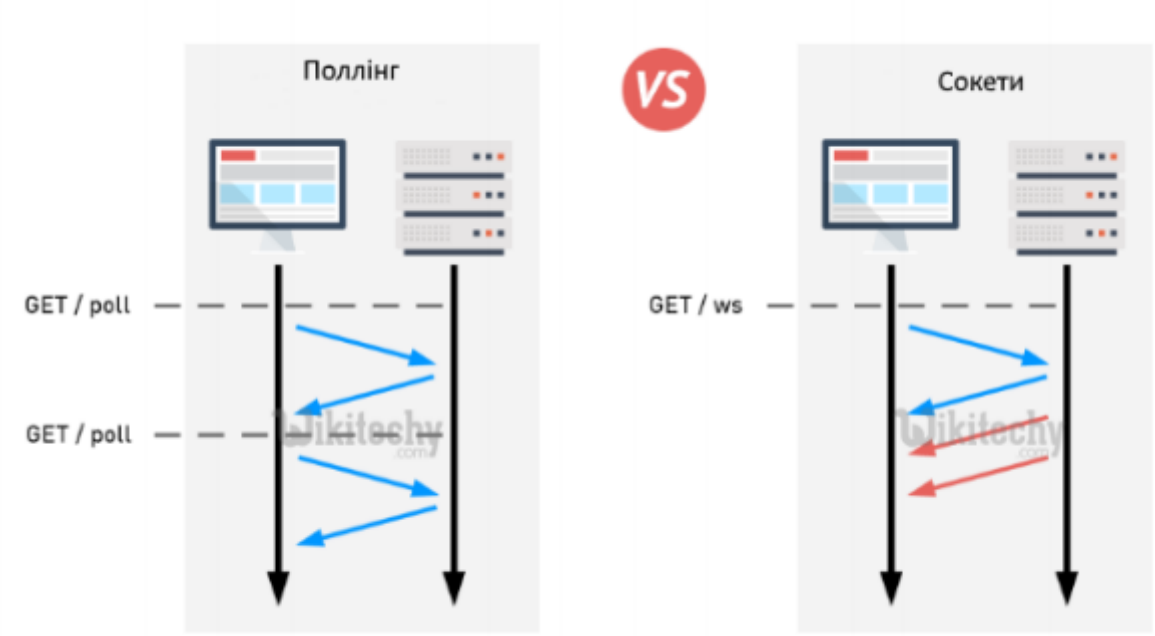

Рисунок 3.16.

брандмауера.

Загальний принцип побудови всіх модулів та їх зв’язку із зображеннями можна побачити на рисунку 3.17.

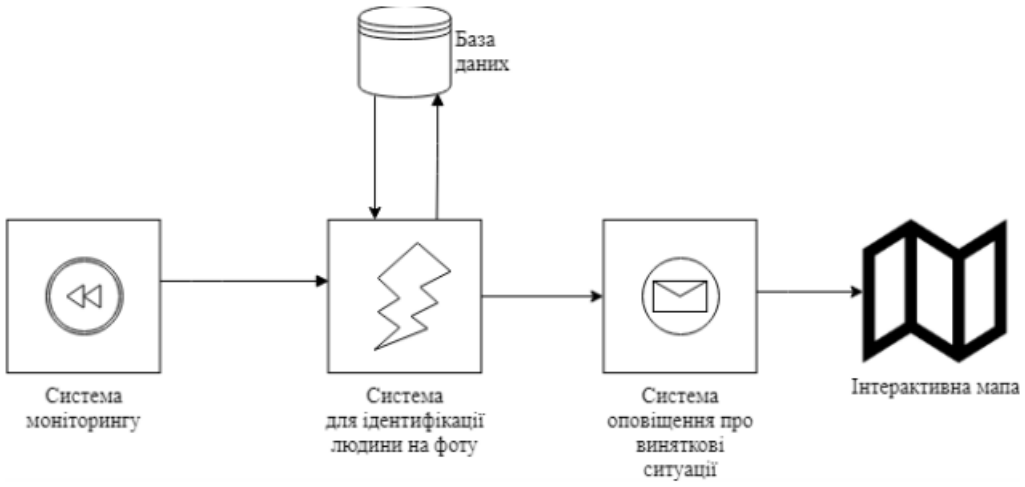

Рисунок 3.17.

За допомогою JSON Web Token (JWT) в системі відбувається вся комунікація, це дуже важливо, так як загалом це відкритий стандарт, який заснований на JSON (RFC 7519) з метою створення маркерів доступу, котрі можуть підтримувати певну кількість вимог. Токени будуть підписані особистим ключем з однієї сторони, в основному сервера, то обидві сторони можуть здійснити перевірку того, що маркер являється правильним. Маркери розроблені компактними, придатними до використання у контексті веб-браузера, URL-безпечними. Слід зазначити, що JWT можна використовувати з метою передавання ідентичності користувачів котрі здійснили аутентифікацію між постачальником послуг та постачальником ідентифікаційних даних (дивитися рисунок 3.18.).

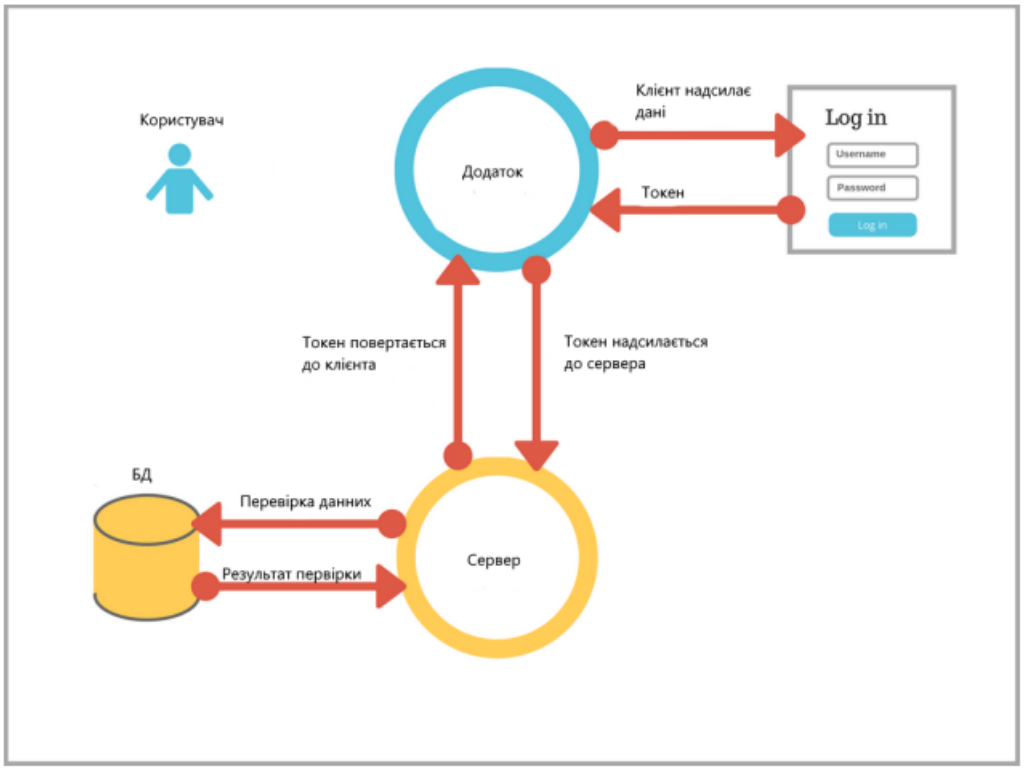

Рисунок 3.18.

При аутентифікації користувача, який успішно увійшов до системи, використавши при цьому свій обліковий запис, маркер JSON повертається та повинен зберегтися локально (дозволяється використовувати файли cookie) окрім традиційного підходу з метою створення сеансу на сервері, а також повернення cookie.

4. ТЕСТУВАННЯ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Тестування роботи

При тестуванні було перевірено основні модулі системи. На рисунку 4.1. показано для створення нового користувача приклад тіла зображень.

Рисунок 4.1.

Результатом даного процесу було забезпечено створення OBM моделі користувача та своєчасно збережено до бази даних (Рисунок 4.2.).

Рисунок 4.2.

Було також перевірено коректність роботи передачі даних, під час тестування модуля сповіщення, про виявляння чи зворотність суб’єкту на знімках та датчиках (дивитися рисунок 4.3.).

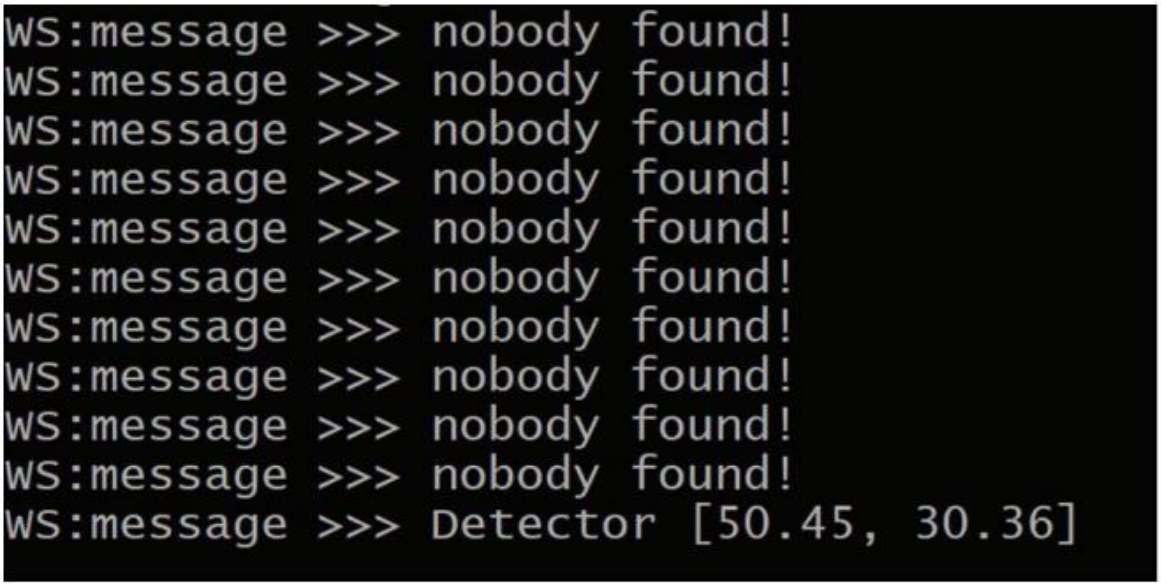

Рисунок 4.3.

При використанні linux-утиліти, з метою навантажувального тестування siege було здійснено вимірювання результативності роботи системи під час оброблення фотоданих та спрацювання сенсорів руху, при цьому, земулювавши 25 одночасних користувачів, які відсилали у середньому 17 запитів на секунду (дивитися рисунок 4.4.).

В середньому за 1,43 секунду система обробляє ці запити при таких умовах, що є доволі непоганим результатом. Було використано віддалений сервер для тестування, а не локальний, що варто підмітити.

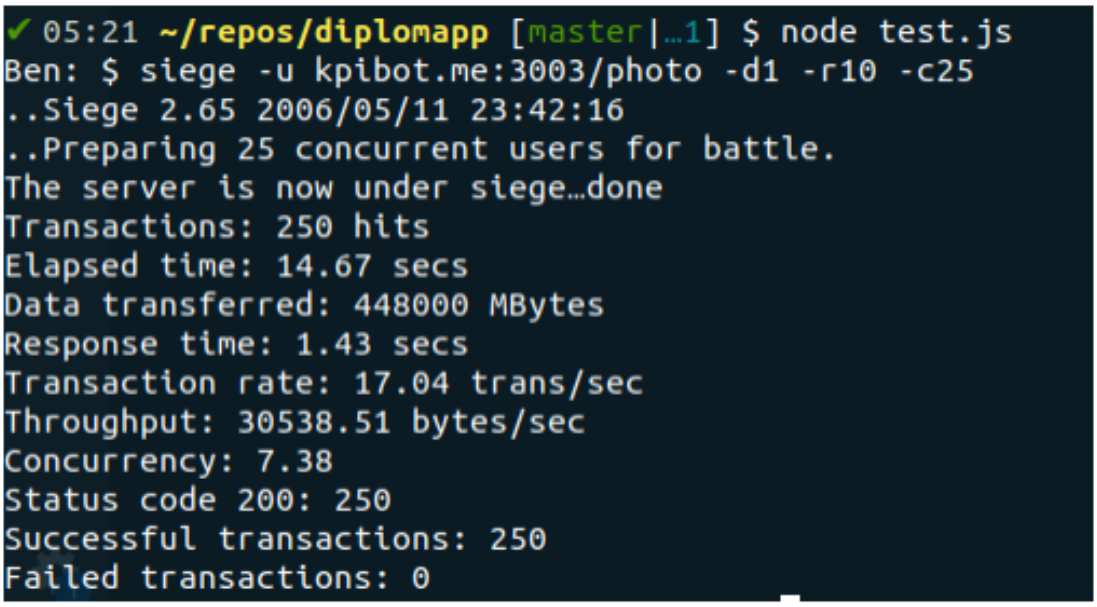
✔ 05:21 ~/repos/diplomapp [master|…1] $ node test.js
Ben: $ siege -u kpibot.me:3003/photo -d1 -r10 -c25
..Siege 2.65 2006/05/11 23:42:16
..Preparing 25 concurrent users for battle.
The server is now under siege…done
Transactions: 250 hits
Elapsed time: 14.67 secs
Data transferred: 448000 MBytes
Response time: 1.43 secs
Transaction rate: 17.04 trans/sec
Throughput: 30538.51 bytes/sec
Concurrency: 7.38
Status code 200: 250
Successful transactions: 250
Failed transactions: 0

Рисунок 4.4.

ВИСНОВКИ

Створення розподіленої системи моніторингу об’єктів охорони при високій швидкості оповіщення являється головною метою даного дипломного проекту. При виконанні роботи було здійснено аналіз, а також розглянуто переваги над класичними підходами та методами, використовуючи при цьому сучасні технології веб-розробки. Це в свою чергу дало можливість подолати досить розповсюджені проблеми, які виникають в існуючих програмних рішеннях.

Було здійснено реалізацію певних баз даних з метою побудови геоінформаційних систем, котрі могли б доволі ефективно використовуватися у галузі охоронних систем, це те, що було в ході розробки. Дана технологія лише розпочинає своє існування, тому передові компанії здійснюють їх розробку та модернізацію.

Для подальшого розвитку даної системи є розширення можливості підключення до неї різних датчиків, а саме звукових та світлових. Оскільки вони також мають доступ до більшості мобільних платформ. З метою забезпечення більшої гарантії, стосовно реагування служб охорони та власника об’єкту слід додавати нові шляхи оповіщення, а також їхнього дублювання. Свого роду: телефонні дзвінки, сповіщення на мобільних додатках, смс-повідомлення, електронна пошта та інше.

В цілому дана розроблена система моніторингу демонструє свою надійність і має повне право на своє існування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Establishing Websocket PUBSUB server with Redis and Asyncio [Електронний ресурс] – 2020 – Режим доступу: https://yeti.co/blog/establishing-a-websocket-pubsub-server-with-redis-and-asyncio-for-the-light-sensor/ – Дата доступу: червень 2020.

2. Американська компанія по стандартизації та сертифікації в галузі техніки безпеки [Електронний ресурс] – 2014 – Режим доступу: квітень 2020.

3. Medium RESTful-api [Електронний ресурс] – 2018 – Режим доступу: https://medium.com/@andr.ivas12/rest-простым-языком-90a0bca0bc78 – Дата доступу: червень 2020.

4. Веб-ресурс охоронної системи на основі AI wave2cloud [Електронний ресурс] – 2015 – Режим доступу: https://www.wave2cloud.com – Дата доступу: травень 2020.

5. Стаття про метод опорних векторів [Електронний ресурс] – 2017 – Режим доступу: http://www.dstu.dp.ua/Portal/Data/74/72/3st13-17.pdf – Дата доступу: квітень 2020.

6. Українська компанія по розробці охоронних систем [Електронний доступ] – 2012 – Режим доступу: https://secur.ua/ajax?gclid=EAIaIQobChMI36CTjcuD6gIVi6sYCh1QKQnbEAAYASAAEgLMCfD_BwE – Дата доступу: травень 2020.

7. Lee, Seungmug; Wilson, Harry. “Spatial impact of burglar alarms on the decline of residential burglary” [Текст] – Palgrave Macmillan, a division of Macmillan Publishers Ltd, 2012 – ст. 85-93.

8. Посилання на RPC стандарт WebSocket [Електронний ресурс] – 2014 – Режим доступу: https://habr.com/ru/post/441854/ – Дата доступу: червень 2020.